NAVAL

POSTGRADUATE

SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

TRACKING SUBPIXEL TARGETS WITH CRITICALLY SAMPLED OPTICAL SENSORS

by

James T. Lotspeich

September 2012

Dissertation Supervisor: Mathias Kölsch

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 01–Sep–2012	**2. REPORT TYPE** Dissertation	**3. DATES COVERED** *(From — To)* SEP 2009–SEP 2012
4. TITLE AND SUBTITLE Tracking Subpixel Targets with Critically Sampled Optical Sensors	**5a. CONTRACT NUMBER**	
	5b. GRANT NUMBER	
	5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Lotspeich, James T.	**5d. PROJECT NUMBER**	
	5e. TASK NUMBER	
	5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Naval Postgraduate School, 1 University Circle, Monterey, CA 93943	**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**	
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)	

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. I.R.B Protocol Number N/A.

14. ABSTRACT

In many remote sensing applications, the area of a scene sensed by a single pixel can often be measured in square meters. This means that many objects of interest in a scene are smaller than a single pixel in the resulting image. Current tracking methods rely on robust object detection using multi-pixel features. A subpixel object does not provide enough information for these methods to work. This dissertation presents a method for tracking subpixel objects in image sequences captured from a stationary sensor that is critically sampled spatially. Using template matching, we estimate the maximum a posteriori probability of the target state over a sequence of images. A distance transform is used to calculate the motion prior in linear time, dramatically decreasing computation requirements. We compare the results of this method to a previously state-of-the-art track-before-detect particle filter designed for tracking small, low contrast objects using both synthetic and real-world imagery. Results show our method produces more accurate state estimates and higher detection rates than the current state of the art methods at signal-to-noise ratios as low as 3dB.

15. SUBJECT TERMS

Subpixel, Tracking, Hidden Markov Model, Viterbi, Distance Transform

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
Unclassified	Unclassified	Unclassified	UU	93	

THIS PAGE INTENTIONALLY LEFT BLANK

Tracking Subpixel Targets with Critically Sampled Optical Sensors

James T. Lotspeich

Major, United States Air Force
B.S., United States Air Force Academy, 1999
M.S., Air Force Institute of Technology, 2003

Submitted in partial fulfillment of the
requirements for the degree of

**DOCTOR OF PHILOSOPHY IN
COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL
September 2012**

Author:

James T. Lotspeich

Approved By:

Mathias Kölsch Associate Professor Computer Science Dissertation Supervisor	Richard C. Olsen Professor Physics
Craig Martell Associate Professor Computer Science	Joel Young Assistant Professor Computer Science
Timothy Chung Assistant Professor Systems Engineering	Duane Davis Assistant Professor Computer Science

Approved By:

Peter Denning, Chair, Department of Computer Science

Approved By:

Douglas Moses, Vice Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In many remote sensing applications, the area of a scene sensed by a single pixel can often be measured in square meters. This means that many objects of interest in a scene are smaller than a single pixel in the resulting image. Current tracking methods rely on robust object detection using multi-pixel features. A subpixel object does not provide enough information for these methods to work. This dissertation presents a method for tracking subpixel objects in image sequences captured from a stationary sensor that is critically sampled spatially. Using template matching, we estimate the maximum a posteriori probability of the target state over a sequence of images. A distance transform is used to calculate the motion prior in linear time, dramatically decreasing computation requirements. We compare the results of this method to a previously state-of-the-art track-before-detect particle filter designed for tracking small, low contrast objects using both synthetic and real-world imagery. Results show our method produces more accurate state estimates and higher detection rates than the current state of the art methods at signal-to-noise ratios as low as 3dB.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

PSF	point spread function
SNR	signal-to-noise ratio
SLAM	simultaneous localization and tracking
EO	electro optical
IID	independent and identically distributed
IR	infrared
DARPA	Defense Advanced Research Projects Agency
ARGUS—IS	autonomous real-time ground ubiquitous surveillance—imaging system
JCS	Joint Chiefs of Staff
LEO	low earth orbit
IED	improvised explosive device
UAV	Unmanned aerial vehicle
UAV	unmanned aerial vehicle
IFOV	instantaneous field of view
FAA	Federal Aviation Administration
NAS	national air space
SAR	synthetic aperture radar
LIDAR	light detection and ranging
FOV	field of view
RMS	root mean squared
PF	particle filter
TBD	track before detect
MCMC	monte carlo markov chain
CONDENSATION	conditional density propagation for visual tracking
pdf	probability distribution function
SIR	sequential important resampling
GPU	graphics processing unit
RMS	root mean squared

PETS	performance evaluation of tracking and surveillance
TP	true positive
TN	true negative
FP	false positive
FN	false negative
TG	total ground truth frames
TF	total number of frames
MRF	markov random field
CRLB	Cramer-Rao lower bound
MGE	modifying greedy exchange
GOA	greedy optimal assignment
GPC	greedy point correspondence
DoE	design of experiments
RSM	response surface methodology
AEE	average estimation error
EER	estimation error rate of change
TEE	total estimation error
MEE	median estimation error
FWHM	full width at half-maximum
RADAR	radio detection and ranging
EEV	estimation error variance
NCV	nearly constant velocity
MAP	maximum a posteriori probability
SIFT	scale-invariant feature transform
SURF	speeded up robust feature
DoG	difference of gaussians
GLOH	gradient location and orientation histogram
IRST	infrared search and track
ALRT	approximate likelihood ratio test
MM	multiple-model
AMM	autonomous multiple-model

CMM	cooperating multiple-model
VSMM	variable-structure multiple-model
CA	constant acceleration
GPBn	generalized pseudo-Bayesian algorithms of order-n
EKF	extended Kalman filter
UKF	unscented Kalman filter
AWGN	additive white Gaussian noise
ATC	air traffic control
HIFOV	horizontal instantaneous field of view
VIFOV	vertical instantaneous field of view
IFOV	instantaneous field of view
MoG	mixture of Gaussians
PCA	principal component analysis
EM	expectation maximization
ICA	independent component analysis
INMF	incremental non negative matrix factorization
IRT	incremental rank (R_1, R_2, R_3) tensor
ML	maximum likelihood
HMM	hidden Markov model
KDE	kernel density estimation
VLO	very low observable
PMV	pixel-matched Viterbi
PMV—DT	pixel-matched Viterbi with distance transform
CCI	central composite inscribed
CCD	charge-coupled device
CMOS	complementary metal-oxide-semiconductor
GMPF	generalized matched-pixel filter
RMSE	root mean squared error
TDR	tracker detection rate
FAR	false alarm rate
OTE	object tracking error

DT	distance transform
MMSE	minimum mean-squared error
SCR	signal-to-clutter ratio
MWIR	mid-wave infra-red
PWM	pulse width modulation

EXECUTIVE SUMMARY

In many remote sensing applications, the area of a scene sensed by a single pixel can often be measured in square meters. At this resolution, small targets of interest present a surface area to the sensor that is smaller than the total area covered by the pixel. This results in the pixel's sensed intensity comprising both the target's intensity and the background intensity in the area covered by the pixel. This means that subpixel sized objects will not present texture or edge features to the sensor, but only intensity features, and those features are mixed, in proportion to target surface area and pixel resolution, with the intensity of the background scene.

Current state-of-the-art tracking methods rely on robust object detection using multi-pixel features such as edges, points, and textures. A subpixel object does not provide enough information for these methods to work. For applications from space, a subpixel target could be something as large as a Somalian pirate boat when imagery is taken from a high orbital altitude. With no method to track such small targets the only solution is to increase the focal length of the system, reducing the field of view of the sensor and increasing weight and complexity.

This dissertation describes a method for tracking subpixel targets using the diffraction of light at the aperture of the sensor as a signal. Based on diffraction, templates are created representing the target's expected projection in the image. These templates are compared to images from the sensor to develop a likelihood of target location. Using this likelihood and a target motion model, the maximum a posteriori probability of the target's state is estimated over the sequence of images. The final result is a path representing the most likely sequence of locations for the target given the imagery and motion model.

The performance of this tracker is tested using synthetic, quasi real-world, and real-world imagery. Results are compared to the current state of the art method for very low observable target estimation. Results show that our method outperforms the state-of-the-art at all signal-to-noise ratios down to 4dB. Additionally, greater than 80% of all estimates are within one pixel of the actual target position down to 3dB signal-to-noise, while the state of the art method achieves greater than 80% accuracy at 6 dB.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

One can only display complex information in the mind. Like seeing, movement or flow or alteration of view is more important than the static picture, no matter how lovely.
– "Epigrams in Programming", by Alan J. Perlis

To my loving wife, who patiently listened to my ramblings every evening and even more patiently acted as my lab assistant when needed. Thank you.

To my committee, whose professionalism and enthusiasm propelled me to work harder and set my sights higher than I had ever thought possible. You have been an inspiration to me. Thank you for your hard work and advice, and your willingness to put in the extra effort to help me finish on time.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

As military use of satellite and unmanned aircraft intelligence increases, the problem of developing actionable intelligence from sensor data is quickly becoming intractable. In January 2010, LtGen David Deptula, Air Force Deputy Chief of Staff of Intelligence, Surveillance, and Reconnaissance said "We're going to find ourselves in the not too distant future swimming in sensors and drowning in data" [1]. Gen Cartwright, Vice Chairman, Joint Chiefs of Staff (JCS) estimates that 2,000 intelligence analysts are needed for each Predator drone fitted with next generation sensors [2]. As new advanced sensors, such as the Defense Advanced Research Projects Agency (DARPA) autonomous real-time ground ubiquitous surveillance—imaging system (ARGUS—IS) sensor, are deployed this number is expected to increase.

Satellite imagery offers a wide-area view that is critical for developing intelligence data. As a satellite passes over an area of interest, a high-resolution camera can be employed to provide a real-time visual of a section of the earth's surface. Due to the height of the satellite, a trade-off must be made between detailed imagery for a very short period of time, or much less detailed imagery for a longer period of time. For example, the highest reported commercial optical resolution for a satellite in low earth orbit (LEO) is 0.41m or 16in [3]. Given a human target at nadir with shoulder width of ~21in, this means that the target projection would occupy some parts of four different pixels of the image at the best possible resolution. If the same satellite were moved to a higher orbit to "stare," the surface area covered by a single pixel increases dramatically, rendering the same human target as a small contribution of the total photon flux sensed by that pixel. While the higher altitude satellite can continuously view the same location, the lower satellite will only be able to gather more detailed information for a short period of time.

Sub-pixel tracking of satellite imagery can be used to solve a number of commercial and military problems. For example, tracking allows planners to analyze tracks of pedestrians in open air locations. These tracks could be used to aid in planning pedestrian paths, crowd control measures, or evacuation routes. Military analysts can use tracking to retrace the paths of any pedestrian targets who approached an area where an improvised explosive device (IED) was recently detonated.

Another advantage of subpixel tracking is reduced bandwidth usage. Tracking can be performed using lower resolution imagery to reduce transmission costs while still allowing analysis currently requiring much higher resolutions. Using the subpixel estimated path, more detailed information can be requested for the parts of the image that are of interest. This same method can be applied on-board the sensor to determine which sections of an image to store, and which to throw out.

Unmanned aerial vehicle (UAV) technology can also benefit from sub-pixel tracking. Current Federal Aviation Administration (FAA) regulations do not allow unmanned aerial vehicles (UAVs) to be used in the national air space (NAS) unless they can perform “see and avoid” to the same level as a human pilot. This means that aircraft must be able to visually detect and track other traffic with the same level of ability as a human pilot. This means that an aircraft on a collision course must be detected and tracked through the aircraft’s ego-motion. For non-cooperating aircraft below 250 knots, the majority of traffic must be detected at 6167ft in order to maneuver to avoid a collision [4]. Given this distance, the instantaneous field of view (IFOV) (i.e., the field of view, in radians, of a single detecting element of the sensor) required to detect the frontal area of the smallest expected traffic is 0.56 milliradians. On a standard electro optical (EO) camera with a field of view (FOV) of 60° and 3,648 by 2,736 pixels (10 MP), the IFOV is 0.28 milliradians. This means that a see and avoid solution must be able to detect and track aircraft occupying no more than 2 pixels of a 10MP image in order to avoid them. If a larger margin of safety is required, the distance must be increased, reducing the number of pixels occupied by the target. A sub-pixel tracking solution makes this problem easier to solve provided that it is robust to error and has a very low false positive rate.

This dissertation outlines the methods used to perform subpixel tracking for critically sampled, additive Gaussian noise optical sensors. Chapter II discusses prior work in the fields of tracking and subpixel target detection. Chapter III presents the methodology used in this dissertation to accomplish the goal of tracking subpixel targets in imagery. The experiments used to validate this methodology are detailed in chapter IV and the results of those experiments are presented in Chapter V. Finally, Chapter VI discusses the implications of the our experimental results, and outlines areas for future research.

II. PRIOR WORK

The problem of tracking subpixel targets in critically sampled imagery has not been heavily researched. Similar problem domains, such as very low observable (VLO) targets offer insights into how to proceed, but methods developed for this field are sometimes ill-suited to subpixel targets. The task of tracking targets using noisy sensors requires knowledge of the optical characteristics of the sensor, the visual and kinematic characteristics of the target, and a number of images captured while the target is moving. This chapter discusses prior work that enables the methodology detailed in Chapter III. The major research areas covered in this chapter are target tracking technology, state of the art methods for tracking VLO targets, methods for performing detection of subpixel targets, and background subtraction techniques.

A. TRACKING AS INFERENCE

Tracking is a form of estimation for non-stationary statistical distributions. Bar-Shalom defines estimation as “the process of inferring the value of a quantity of interest from indirect, inaccurate, and uncertain observations” [5]. In order to estimate a target’s path, we must infer both its location in multiple images of a video sequence and the motion that best describes those locations over the entire image sequence. This naturally decomposes the tracking problem into detection, finding features of the target in a single frame, and estimation, estimating the target’s state in each frame. These two problems are often solved iteratively with the solutions from the previous frames used to aid in the solution of the current frame.

The tracking problem can be most generally formulated as

$$\text{State}_{t+1} = f(\text{State}_t, \text{process noise}_t) \tag{1}$$

$$\text{Measurement}_t = h(\text{State}_t, \text{measurement noise}_t) \tag{2}$$

where the function $f(\cdot)$ is the motion model, describing the evolution of the state over a single time step, and the function $h(\cdot)$ maps a known or estimated state to the expected observation value [5]. Both noise values are assumed to have a known statistical distribution which can be either static or dynamic. The process noise results in increased uncertainty of the estimated state, while the observation typically, but not guaranteed, helps to reduce uncertainty. The goal of tracking is to provide the lowest error state estimate possible (in the least squared error sense) given the process and measurement noise distributions.

The continuous distribution for state inference is

$$p(X_{0\ldots t}, Z_{1\ldots t}) = \int p(X_{1\ldots t}|X_0, Z_{1\ldots t})\mathrm{p}(\mathrm{X}_{1\ldots\mathrm{t}})\mathrm{d}X_{1\ldots t} \tag{3}$$

where the random variable representing target state at time k is denoted as X_k, and the random variable for the sensor observation at time k is Z_k. If the probability distribution for X and Z both follow a known distribution with a closed form solution, then exact inference can be performed on the target state.

Under certain assumptions, the optimal posterior of the target state can be calculated exactly. For instance, Kalman filters, are optimal methods for recursively calculating the posterior density provided that it is Gaussian at every time step. In order to achieve this assumption, the motion and sensor models must be linear since a linear function applied to a Gaussian density results in a Gaussian density. While this assumption is restrictive, the Kalman filter is a popular tool that has been applied successfully in a large number of inference applications [6, 7, 8, 9, 10, 11, 12, 13, 14].

In tracking problems, the distribution for random variables X_k and Z_k typically depend on the prior distribution of X_{k-1}. One of the most popular methods for representing dynamic distributions with dependencies on past states is through a Bayesian networks. In tracking, a specific type of Bayesian network, called a hidden Markov model (HMM) is used to represent the time varying distribution over X and Z. This model assumes that random variables follow three assumptions [15]:

1. The Markov assumption: a state, X_k, is independent of all other states in the model given the previous state, X_{k-1}
2. Stationary state transitions: the probability of a particular state transition is the same regardless of the time at which it occurs. For example, a transition from x_1 to x_2 where $x_1, x_2 \in X$ will have the same probability at time k as it does at time j
3. Independent observations: each observation is assumed to be independent of any previous observation.

These three assumptions allow for a simplified solution to Equation 3 conducive to the tracking problem. Tracking using an HMM falls into two major categories: single state inference, and most likely sequence inference [15]. The first is a form of maximum likelihood (ML) estimation on the HMM distribution, while the second is a maximum a posteriori probability (MAP) calculation on the distribution.

1. Maximum Likelihood Estimation

ML estimation can applied to HMMs to calculate the maximum likelihood of a state given observations, or to calculate the maximum posterior distribution of parameters, given a set of observations [5]. In tracking, ML estimation is the most common method used to infer the target state at a given time. This form of estimation has been used for robot motion control [16], navigation [17], pedestrian tracking [18], pose tracking [19], and an array of other applications [20, 21, 22, 23].

In cases where the ML posterior distribution is differentiable, a closed form solution of the ML estimate can be calculated analytically. For discrete or non-differentiable state spaces, the forward algorithm can be used to calculate the state probability at a given time k using observations $z_{1...k}$ [15]. This algorithm relies on the Markov assumption to calculate states recursively from X_1, through X_k without the need to recalculate the entire sequence of states.

$$p(X_{k+1} = x_j | z_{1...k}) = \sum_{x \in X} p(z_{1...k-1}, X_k = x_i) p(z_k, X_{k+1} = x_j | X_{0...k}) \tag{4}$$

The forward algorithm is the underlying mechanism for many of the tracking methods in the literature. For example, in a particle filter, the product of all previous state probabilities is represented by the population of particles prior to resampling. The resampling step is the equivalent of performing the summation in Equation 4.

A drawback to using ML on an HMM is that the estimate at time k is never updated using future observations. This means that a set of estimated positions using the forward algorithm will not be the same as the most likely set of states over all observations. Adding a backwards step to the forward algorithm solves this problem. The new method, called the forward-backward algorithm, solves for $p(X_k|z_{1...t})$ where $0 < k < t$. In situations where the distribution can be calculated in reverse, this method produces estimates that are closer to the optimal state estimates, however, in applications where approximation techniques are required, such as particle filters, the backwards filtering distribution is often difficult to specify.

One attempt at applying the forward-backward algorithm to a particle filter method is Junkun et al. [24]. This method uses a fixed lag in particle positions so that the particle weights can be calculated using future observations to improve the estimate. The overall complexity of the algorithm is $O(tn^2)$. To achieve faster running time, Junkun et al. implement a number of approximations to the filtering distribution such as local linearization, smoothing over only one step instead of more than one, and replacing a probability calculation with a simpler Euclidean distance calculation. Despite these approximations, the smoothed filter achieves root mean squared (RMS) error as low as 0.1 at signal-to-noise ratio (SNR) of 8.3dB.

2. Maximum a Posteriori Estimation

Rather than calculate single state estimates iteratively, the MAP estimate attempts to determine the probability of a sequence of states by selecting the sequence that results in the maximum posterior probability.

$$p(X_{1...t} | z_{1...t}, X_0) = \underset{X_{1...t}}{\operatorname{argmax}}\, p(z_{1...t} | X_{1...t}) p(X_{1...t} | X_{0...t-1}) \tag{5}$$

Like ML, the MAP estimate is optimal as $t \to \infty$ [5]. However, while ML can be performed “live” as observations are made, always providing the most likely current state, MAP requires the entire sequence of observations in advance. This introduces a lag that is often undesirable in real-world tracking applications.

Additionally, MAP is discussed in the literature as an inferior method with regards to estimation quality. For example, Greig et al. [25] analyze the use of MAP in the reconstruction of binary images. The key finding is that MAP does not perform well for the given problem set, however, this has been taken to mean that other methods such as markov random fields (MRFs) or Gibbs sampling are superior to MAP [26, 27]. Likewise, Fox and Nicholls [28] compare a MAP estimator to minimum mean-squared error (MMSE) and find that MAP produces lower quality estimates. As a result, there are relatively few examples of MAP used in the tracking literature.

Recent work by Tran and Yuan [29], Islam et al. [30], and Yan et al. [31] show that the MAP estimate can produce better results than MMSE and ML for certain problem domains. Each of these methods show better results for MAP estimation than achieved by MMSE or ML. This indicates that the general consensus of MAP as a less accurate estimator is based on misinterpretation of the findings in [25, 28]. Those findings claim that MAP performs poorly relative to MMSE for the specific problem domain. Interpretations that state MAP is an inferior estimator must specify the problem domains for which this is true. Problem domains that exhibit maximum modes in the optimum estimate are expected to perform as well as or better than other estimators. Based on the work in [29], MAP can be applied to certain tracking problems with superior results.

If calculating MAP on a discrete state space, the Viterbi algorithm offers an optimal solution to Equation 5 [32]. The Viterbi algorithm is a dynamic programming method for calculating the MAP in $O(tn^2)$ time. The most common use of this algorithm is in the field of decoding convolutional codes [33, 34, 35], but the methodology is applicable to any HMM with discrete state values [36].

A number of modified Viterbi algorithms have been developed for use in different domains. For example, online Viterbi [37] uses a sliding window of observations rather than requiring that all observations are available at once. Additionally, the computation is modified to use constant memory within the window. While this method sacrifices the optimality of the algorithm, it allows Viterbi to be used on an infinite sequence of observations.

Other modifications, such as A* Viterbi [38] and Tree Viterbi [39], attempt to reduce the total number of paths expanded during calculation. Both methods limit the number of nodes searched through the use of graph search techniques. A* Viterbi uses a priority queue and heuristic to determine which state to expand in the trellis. Tree Viterbi, on the other hand, performs a depth-first search of the trellis and uses branch cuts to limit search time. In both cases, the additional overhead of data structures and decision points can overshadow the gains of expanding fewer nodes in cases where hardware vectorization and cheap transition cost calculation accelerate the standard Viterbi algorithm.

Lazy Viterbi [40] attempts to speed up computation of A* Viterbi by limiting the use of the priority queue. This results in Viterbi search with lower bound of $O(1)$ and worst case of $O(tn^2)$. For estimation problems where the observation length is known a priori, and an admissible heuristic exists to guide the expansion, lazy Viterbi has been shown to speed up the A* calculation by as much as 50%. As with A* Viterbi, lazy Viterbi requires more implementation overhead. Depending on the cost of computing transitions and expanding nodes, the standard Viterbi algorithm can sometimes outperform the lazy method due to vectorization and compiler optimizations.

B. VERY LOW OBSERVABLE TARGET TRACKING

Very little data exists in the literature regarding subpixel target tracking. In order to compare tracking results with published data, it is necessary to look to similar challenging tracking scenarios that have received more attention. The VLO tracking field is similar to subpixel target tracking in many ways. A subpixel target exhibits low SNR in a given frame, while the VLO problem specifies very low SNR as the major obstacle to tracking. Additionally, a subpixel target provides very few pixels of data relative to the overall scene. While VLO tracking assumes that targets are larger than a pixel, most applications focus on targets between 3 and 10 pixels across the largest dimension. This very low pixel count and low SNR makes tracking a difficult problem. These same difficulties exist in the subpixel problem domain to a greater degree. Given the similarity between VLO and subpixel domains, we review the VLO methods for possible application to the subpixel problem domain.

A VLO target is one that has a very low peak SNR in the sensor data. While target size is not necessarily specified in VLO applications, small targets are typical due to their low overall photon flux relative to a larger target. To reduce noise, many tracking algorithms threshold the data so that sensor returns below a certain intensity are not considered by the tracker. Since VLO targets have intensity values that are close to or below the expected noise values, it is usually not possible to achieve acceptable detection probabilities without incurring high false alarm rates. These high false alarm rates make tracking much more difficult.

In addition to low detection rate, VLO targets are also affected by background clutter. Clutter in an optical sensor consists of background objects in the scene such as buildings, roads, trees, or signs and is considered separately from sensor noise. A number of methods have been proposed to increase the signal-to-clutter ratio (SCR) in infrared imagery such as morphological filters [41, 42], wavelet methods [43] and expectation maximization (EM) [44]. Background subtraction methods in computer vision can also be applied to reduce SCR by modeling and subtracting a structured background from a sequence of images. Section D discusses background subtraction methods and the advantages and disadvantages of these methods.

A particular class of filters designed to track VLO are the track before detect (TBD) filters. These filters focus on very small, low SNR targets down to, but not smaller than, a single pixel and typically incorporate the point spread function (PSF) in their detection methods [45, 46]. This class of filters is considered to be the state of the art for VLO tracking methods.

The TBD filter attempts to track a target using multiple detection probabilities over time rather than tracking detections above a given threshold. This allows the filter to integrate motion information over time with multiple detection likelihoods in order to build a better estimate of a target's location. Since the sensor model is non-linear, TBD methods use a particle filter algorithm to perform estimation [47]. Theoretical lower bound performance of these filters indicate the ability to perform with subpixel accuracy, however, while many applications achieve subpixel estimation accuracy, most applications in the literature fail to achieve the lower bound performance [45, 48]. Since TBD filters use particle filter methods, they tend to be relatively fast, with computation time proportional to the number of particles used. The number of particles, however, also relates directly to the ability of the filter to converge on the correct estimate. As the number of dimensions or the size of the domain for any dimension increases, the number of particles required increases [49].

The drawback of using TBD methods is that filter performance degrades rapidly as the SNR falls below 7dB. As demonstrated by Rutten et al. [47], the TBD method produces RMS position error of no less than one pixel at 6dB, and no less than three pixels at 3dB. In contrast, the Cramer-Rao lower bound (CRLB) predicts the theoretical performance of the estimator to be better than 10^{-1} pixels at 4dB [48]. While TBD methods perform close to the the lower bound above 7dB, their performance suffers below this level.

C. SUBPIXEL TARGET TRACKING

Samson et al. address the detection of subpixel targets in a fixed scene [50]. They provides a method to detect subpixel point targets using the sensor's PSF as an identifying characteristic. Using matched filtering theory, a measure is defined to determine the likelihood that a given pixel in an image contains a subpixel object. This measure is used to perform a detection decision by applying a threshold to the likelihood values. The Neyman-Pearson criterion to is used to determine the optimal threshold value for a given false positive rate. The resulting detector is shown to be greater than 70% accurate for images with SNR=14dB.

The primary contribution of Samson et al. is the generalized matched-pixel filter (GMPF). This filter calculates the estimate of the unknown target photon flux, α, and returns the minimum sum of squared error estimate for the template centered at each pixel in an image. The GMPF is calculated as:

$$\ell(z^{(x,y)}) = \frac{\left|(s^{\varepsilon_0})^T R^{-1} k(z^{(x,y)})\right|^2}{(s^{\varepsilon_0})^T R^{-1} s^{\varepsilon_0}} \tag{6}$$

The variable s^{ε_0} is a 9x1 vector representation of a 3x3 pixel template. This template is generated by calculating the ensquared energy of each pixel assuming a target with photon flux α is located in the center of the template. This template is calculated by convolving the sensor's PSF with a Dirac delta function representing a point target with intensity α. The variable R represents the sensor's noise covariance matrix, and $k(\cdot)$ is a function that returns a 9x1 vector representation of the 3x3 pixel patch with center pixel of (x,y) in the observation z. Since the sensor noise is assumed independent and identically distributed (IID), R becomes $I^9 r$ where I^9 denotes an identity matrix with diagonal of size 9.

The GMPF provides the likelihood that a subpixel target exists in any given pixel. It does not provide an estimate of the subpixel location of that target. Since GMPF assumes a template with the target located in the center of a pixel, objects offset from the center will produce a lower overall likelihood. In order to calculate the likelihood of a pixel in any subpixel position, Samson et al. propose an approximate likelihood ratio test (ALRT) that uses the trapezoidal rule to approximate the integral:

$$\mathscr{L}(x,y) = \int_{-0.5}^{0.5} \int_{-0.5}^{0.5} \frac{1}{(s^{(i,j)} R^{-1} s^{(i,j)})^{0.5}} \exp\left(\ell(x,y)\right) \mathrm{d}i \mathrm{d}j \tag{7}$$

While the ALRT method provides a measure of pixel-wise likelihood, it does not provide an estimate of subpixel location. To determine subpixel location, a second ML estimate must be performed for each pixel likelihood exceeding the specified threshold. A large number of pixels falling above the threshold limit causes a high computational load. Additionally, the subpixel position estimate does not incorporate any prior location probabilities or data from multiple observations over time. In order to use this method for tracking, it is necessary to specify a tracking method that can use detections provided by the ALRT method.

D. BACKGROUND SUBTRACTION

The background of a scene consists of all of the objects that are not necessary for tracking the target of interest. Background objects may be moving, but are generally assumed to be stationary with only small movements relative to the motion of the target. To eliminate spurious clutter in the estimation process, it is advantageous to subtract the background from the scene before performing inference.

A variety of background subtraction techniques can be found in the literature. For this investigation, we focus on techniques that assume a mostly static background with the majority of changes between images occurring as the result of foreground objects moving through the scene. This is based on the assumption that most targets of interest will move with a speed that much higher than the surrounding background. For example, changes in illumination, cloud cover, and texture over a sequence of images will typically be much slower over a given time window than the changes caused by a vehicle or person moving through the scene.

According to Cheung and Kamath, background subtraction typically follows the steps of preprocessing the image, modeling the background, detecting foreground objects, and data validation, with foreground masks as the final outcome [51]. Bouwmans labels these steps as modeling, initialization, maintenance, and foreground detection [52]. For each background subtraction technique, the main concerns that must be addressed are large-scale changes in illumination over time, shadows cast by moving objects, illumination changes due to environment such as snow, rain or cloud cover, and objects that merge into the background for a time, then return to the foreground (e.g., a car that parks for a period of time could be considered background until it pulls out and starts moving again). Each of these problems can be addressed using a number of methods with varying strengths and weaknesses. The challenge is to determine which method provides the proper balance of computational complexity and accuracy under expected conditions.

For the subpixel tracking problem, we assume a stationary, monochromatic sensor. Images consist of luminance values only. The targets we wish to track are generally slow moving, that is, less than a pixel of motion per frame, with a time delay between images that is sufficient to capture the target motion. Due to bandwidth and utilization constraints, it is not possible to obtain large numbers of images of the same scene, so methods that use relatively few images (<100) in a sequence are desirable.

Bouwmans et al. identify 17 situations that cause basic background modeling techniques to fail [52, 53, 54]. Of these situations, the ones of greatest concern in our application are noise, gradual illumination changes, bootstrapping (initialization or training on the same set of images that subtraction is required for), camouflage (objects with similar color or texture as the background), and multimodal background. Using these situations, methods proposed in the literature to deal with one or more of these methods were reviewed for use in our application.

1. Basic Background Subtraction

The simplest method of background subtraction is frame differencing. An image at time t is subtracted from an image at time $t-1$, and any pixel with a value greater than some threshold, τ, is labeled as foreground. This method is fast, requires little memory, and can be performed in real time. On the other hand, selection of foreground objects is very sensitive to the selection of τ. It must be set sufficiently high to filter out sensor noise, but in cases of shadows or highlights, this high setting could cause errors. This method also registers small background movements as foreground, such as leaves blowing in the wind, wave action, and cloud movement.

The running Gaussian average proposed by Wren et al. is only slightly more complex than frame differencing [55]. This method calculates the per-pixel intensity distribution, $\mathcal{N}(\mu_t, \sigma_t)$, over a period of k frames. A pixel is labeled as foreground if

$$|I_t - \mu_t| > \gamma\sigma_t \tag{8}$$

where I_t is the image at time t and γ is a parameter chosen based on the desired sensitivity of the model, typically set to 2.5. The model is updated based on a weight, α, that determines how quickly μ and σ

changes based on new observations. This so-called adaptation rate allows the model to be tuned for fast or slow background dynamics. A high value for α incorporates slow moving objects into the background model while a low value more likely classifies small changes as foreground for a large number of frames while the model adjusts to incorporate those changes. The advantages of this method are very low memory and computation requirements. This method addresses noise and gradual illumination changes much better than simple frame difference. Since the values of α and γ are fixed *a priori*, fluctuations in the background result in increased misclassification. Additionally, waking and sleeping foreground objects, as well as shadows and highlights produce high error rates unless α is set to allow for rapid adaptation. A high adaptation rate, however, results in slow-moving foreground objects being classified as background.

Lo and Velastin recommend using the median instead of the mean for Equation 8, arguing that it results in a closer model match than the mean. Likewise, Cucchiara et al. claim that the median value can be used to construct a background model using a subsampled frame up to a factor of 10 times smaller than the original frame [56]. While the median filter produces better results, it has the same disadvantages as the mean filter with the added disadvantage that the median cannot be easily calculated as a running median without storing all n previous frames. Additionally, the use of σ_t is no longer a statistically rigorous test relative to the median, so a fixed threshold τ is used instead.

To address the cost of median filter while maintaining the superior performance relative to the running mean, McFarlane and Schofield [57] propose an approximate median method. The approximate median is initialized by setting the first image in the sequence to the median image. For each new image, the median image is updated by incrementing or decrementing any pixel in the median image that is less than or greater than, respectively, the corresponding pixel of the current image. This results in an approximation of the median that can be calculated on the fly. Results reported in [52, 57] show that the approximate median filter performs similarly to the median filter at a substantial saving in memory usage.

2. Statistical Background Subtraction

One of the main disadvantages of these basic filters is that they assume a single mode distribution of pixel intensities. While this assumption holds for many pixels in a scene, it is possible for multiple background objects to appear in the same pixel over time. For example, a cloud will have one intensity distribution, while the ocean surface beneath the cloud will have a different distribution. While the basic methods will adapt to changes in cloud cover over time, it is also possible to use a mixture of Gaussians (MoG) to represent the background intensities. The first use of the MoG method was proposed by Stauffer and Grimson [58]. Power and Schoonees reformulate Stauffer and Grimson's work with several corrections and discuss the theoretical underpinnings of the MoG method [59].

The basic MoG method uses multiple Gaussian distributions to model both foreground and background objects. A mixture is initialized for each pixel using EM and a predetermined number of Gaussians, K (usually between 3 and 7). Each Gaussian is assigned a weight, ω_k, and parameters $\theta_k = \{\mu_k, \sigma_k\}$. For a given image, the closest matching Gaussian, $\hat{k} \in K$, for each pixel is chosen as

$$\hat{k} = \underset{k}{\operatorname{argmax}} \left(\omega_k p(Z|k, \theta_k) \right) \tag{9}$$

Z is the sensor observation.

Once the current pixel state is estimated, the foreground distributions must be segmented from the background distributions. Since the MoG estimates both distributions, Stauffer and Grimson provide a fast procedure for performing segmentation based on the ratio ${}^{\omega_k}/_{\sigma_k}$. Background distributions are expected to occur frequently; therefore, having a higher ω_k value, and they are not expected to vary dramatically so a lower σ_k value is expected. Based on these assumptions, Stauffer and Grimson rank the states for each pixel using the ratio criterion. The first B ranked states with accumulated weight greater than a threshold, τ, are selected as background distributions with all others labelled as foreground.

$$B = \underset{b}{\operatorname{argmin}} \left(\sum_{k=1}^{b} w_k > \tau \right) \tag{10}$$

To maintain the pixel-wise distribution mixtures over time, the model is updated at each time step using:

$$\hat{\omega}_k = \frac{1}{N} \sum_{t=1}^{N} p(k|X_t, \Phi) \tag{11}$$

$$\hat{\mu}_k = \frac{\sum_{t=1}^{N} X_t p(k|X_t, \Phi)}{\sum_{t=1}^{N} p(k|X_t, \Phi)} \tag{12}$$

$$\hat{\sigma}_k^2 = \frac{\sum_{t=1}^{N} ((X_t - \hat{\mu}_k) \circ (X_t - \hat{\mu}_k)) p(k|X_t, \Phi)}{\sum_{t=1}^{N} p(k|X_t, \Phi)} \tag{13}$$

where $\circ$ denotes element-wise multiplication and $\Phi = \{\omega_1 \ldots \omega_K, \theta_1 \ldots \theta_K\}$.

To increase the speed of their algorithm, rather than calculate the exact value for $p(k|X_t, \Phi)$ in eqs 11, 12, and 13, Stauffer and Grimson define a match to be any value within $\lambda\sigma$ of the mean of the one of the distributions with $\lambda = 2.5$ as the recommended value. Power and Schoonees recommend using hysteresis thresholding instead by designating an upper (λ^+) and lower (λ^-) threshold where pixels that lie between the two values are labeled background or foreground based on an 8-connected morphological matching function [59].

The MoG approach to background modeling has proven very popular in the literature with a large number of modifications to the original algorithm. For example Kim et al. attempt to characterize the background model using Laplacian distributions rather than Gaussians [60]. A number of papers have attempted to overcome a fixed value of K Gaussians through estimation [61], simulation [62], and hill climbing [63, 64]. To improve model initialization in sequences where a "clean" background cannot be easily obtained, researchers have applied EM [65], approximate EM [66], [67], QR decomposition [68], and optic flow [69].

Another area of intense research is the adaptation rate of the algorithm. In its original embodiment, the adaptation rate is statically set at the beginning of a run. In order to deal with rapid scene changes, illumination changes, or poor initial model estimation, the adaptation rate must be very fast. On the other hand, high adaptation rates result in foreground objects being incorporated into the background model if they do not change rapidly enough. To address this problem, the idea of using foreground counters has been proposed in several different settings [67, 70, 71]. While each method differs in implementation, the core principle exploited is to limit the amount of time a foreground object can stop before it is subsumed into the background model. This allows for a slow adaptation rate for illumination changes while selectively increasing the adaptation rate for foreground objects that stop moving for a period of time.

MoG estimation has two major drawbacks: initialization priors and adaptation rates. In surveillance applications where it is relatively easy to obtain images free from foreground objects, these images can be used to initialize the model. For situations where it is difficult or impossible to obtain a sufficient number of "clean" background images, the algorithm exhibits large error rates during initial foreground/background segmentation with results improving over time. For remote-sensing applications, it is possible to develop background models using images taken over multiple passes of the target region, however, it is generally impossible to guarantee target-free images.

The adaptation rate for MoG must be slow enough that foreground objects are not subsumed into the background model. This creates a limit to how well the background can be matched with a small number of frames. While the number of frames in a test sequence is rarely discussed in the literature, a demonstration video released by Power and Schoonees contains hundreds of frames in the sequence. If we are limited in the number of frames we can capture, for instance one image per orbit, it will be costly to collect enough images for the background model to be build with sufficient accuracy.

Building on the popularity and performance of MoG background subtraction, Elgammal et al. developed kernel density estimation (KDE) as a nonparametric method of estimating the background distributions [72, 73, 74]. In KDE a histogram of pixel intensities is constructed over a number of frames. This histogram is then convolved with a kernel, usually Gaussian, with a set variance, h, called the kernel bandwidth. The resulting distribution is used in a similar manner to MoG to segment background and foreground pixels. Since it uses a histogram representation for each pixel, KDE does not require predetermined or adaptive values for

K. It also does not require estimation of μ_k or σ_k. These advantages are offset somewhat by the need to use a set bandwidth size, h. As for MoG, KDE requires a large number of frames to develop a good background model. It also requires special selection criteria for bootstrapping applications.

E. MEASURES OF PERFORMANCE

Methods of measuring estimation quality are necessary to validate the methodology presented in Chapter III. The literature provides a number of methods for comparing the tracking algorithm performance. Ristic et al., and Bessel use RMS error of each state variable relative to the ground truth state [45, 75]. Kalal and Wu, on the other hand, present real-time error detection methods that attempt to solve the tracking problem from the current time step to the initial distribution $P(X_0|Z_0)$ without the need to compare to ground truth data [76, 77]. As part of the performance evaluation of tracking and surveillance (PETS) workshop, Bashir and Porikli discussed a suite of metrics for evaluating the performance of various tracking methods [78]. This suite provides a number of standardized evaluation metrics covering both frame-by-frame and total path performance.

The metrics used in [78] are based on a bounding box that fully encloses the target's footprint in the image. By using a bounding box, a simple test of estimation quality is to determine whether or not an estimation position falls within the ground truth bounding box. Using this bounding box method, Bashir and Porikli evaluate the count of true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs) on a frame-by-frame basis. Using these metrics, it is possible to construct sequence-wide measures of quality such a tracker detection rate (TDR), specificity, accuracy, false alarm rate (FAR), and a number of other measures.

In addition to frame-based metrics, [78] introduces object-based metrics. These metrics use the entire estimated path of an object as opposed to individual frame-wise estimation points. For example, the path-based version of TP uses the mean Euclidean distance between the ground truth path and the estimated path for all overlapping time steps to determine whether or not a TP has occurred. As with the frame-based methods, Bashir and Porikli define the metrics TP, TN, FP, and FN for objects. Using these metrics, TDR and FAR are reported on a path-wide basis. Additionally, the metric object tracking error (OTE) is calculated as the RMS distance between the estimated object location and the bounding box centroid. The metrics developed in [78] provide a starting point for evaluating the quality of various tracking methods. Section A discusses modifications made to these methods to adapt them for subpixel tracking scenarios.

F. CONCLUSION

The literature provides a number of methods that can be applied to the subpixel problem. While many of these methods do not apply directly to the problem domain, with small modifications, tracking techniques such as MAP inference using the Viterbi algorithm can be adapted to work with subpixel targets. The methods developed in Samson et al. [50] are particularly useful for the development of a likelihood function to apply to the MAP estimation. The various background subtraction methods, while not developed specifically for subpixel foreground objects, provide a means to remove spurious background intensities from the image, making the tracking task somewhat easier. Finally, metrics developed by Bashir and Porikli offer a starting point for measuring performance of the methods developed in Chapter III.

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY

This chapter details the pixel-matched Viterbi (PMV) algorithm for tracking subpixel targets. Sections A and B define the target and sensors models used to calculate the target state likelihood. Section C discusses the motion model used for the experiments in Chapter IV. In Section 1, the Viterbi algorithm is defined for calculating the maximum a posteriori probability (MAP) of the target states given the target, sensor, and motion models. Finally, the distance transform algorithm used to achieve linear time complexity for the Viterbi algorithm is explained in Section 2.

A. TARGET MODEL

A target consists of any object of interest smaller than the size of a single pixel. The target is assumed to present a surface area to the sensor, as well as to reflect or emit some amount of radiation in the sensor's spectral band. The amount of surface area and spectral response, that is, the amount of spectral energy reflected or emitted by the target at each point on its visible surface, is assumed to be unknown. The target's intensity contribution, α, is determined by integrating the target's spectral response over the total visible surface area. Ignoring the affect of the sensor point spread function (PSF), the contribution of the target to a given pixel intensity is the sum of the background's total non-occluded intensity, and the target's total intensity, $\alpha + B$. Since the target is subpixel, the value of α is the only target parameter measured by the sensor. In fact, since both target visible area and spectral response are unknown, it is not possible to determine the size or spectral response of a subpixel target without making further assumptions about the target. Since a target area cannot be determined using intensity alone, it is possible without loss of generality to specify the target as a point source total intensity, α.

A target's signature is defined as

$$T(x,y) = \alpha\delta(x,y) \tag{14}$$

where $\delta(x,y)$ is the Dirac delta function at location (x,y).

B. SENSOR MODEL

A sensor consists of a set of sensing elements, called pixels, arranged in a grid format, such as a charge-coupled device (CCD) or complementary metal-oxide-semiconductor (CMOS) device. Sensor spectral range is used to determine the size of the PSF, and is easily specified for different sensor types. This means that the methods discussed in this chapter can be applied to infrared (IR), radio detection and ranging (RADAR), or standard optical sensors provided that the sensing elements are designed to determine the photon flux at each pixel site during a fixed integration interval, and the sensor exhibits a PSF due to aperture affects.

To use the sensor's PSF as a target signature, the sensor must be critically sampled or oversampled. Since a critically sampled sensor represents the smallest number of pixels that can be used to accurately sample the PSF, the remainder of this work assumes a critically sampled sensor. As discussed in Chapter II, this means that the size of each photosite is calculated as the Nyquist rate necessary to sample the full bandwidth of the sensor. The maximum sensor bandwidth is determined by the PSF. According to the Rayleigh criteria, the maximum spatial frequency that a sensor can sample is determined by the distance from the maximum intensity point of the PSF to the first zero ring [79]. For an Airy disk, this distance is $1.22\lambda N$, where λ represents the mean of the sensor's spectral frequency range, and N is the ratio of aperture diameter to focal length. According to the Nyquist sampling theorem [80], the optimal sampling rate, r_c, for the PSF $2.44\lambda N$. By setting the size of the sensor photosite to $^1/_{r_c}$, the sensor is critically sampled. It is common practice to design satellite and optical sensors for remote sensing applications using this criteria [81].

1. Approximating the Point Spread Function

The PSF of a sensor with a round aperture is an Airy disk. This function is calculated using the Bessel function of the first kind [79].

$$\mathrm{PSF}(\theta) = I_0 \left(\frac{2J_1(ka\sin\theta)}{ka\sin\theta} \right)^2 \tag{15}$$

The variable k is $^{2\pi}/_{\lambda}$, a is the diameter of the aperture, and θ is the angle between a line drawn perpendicular to the aperture through the aperture center, and the line drawn from the center of the aperture to the observation point. The term $ka\sin\theta$ can be rewritten as $^{\pi q}/_{\lambda N}$, where q is the radial distance on the sensor plane from the line drawn perpendicular to the aperture through the aperture center. Figure 1 shows the relationship of these values.

In Section 3, we describe the generation of templates for matching the target signal. A template is determined by calculating the total area under the PSF that is covered by a single pixel. In order to calculate these templates quickly, we approximate the PSF using a Gaussian function. The PSF is a a rapidly decaying sin function with full width at half-maximum (FWHM) of $1.028\lambda N$ where λ is the mean of the sensor's spectral bandwidth, and N is the ratio of focal length to aperture diameter. While the tails of a Gaussian do not fall to zero like the PSF, a Gaussian with FWHM equal to that of the PSF is a close approximation. The FWHM of a Gaussian occurs at

$$\mathrm{FWHM} = \sigma 2\sqrt{2\ln 2} \tag{16}$$

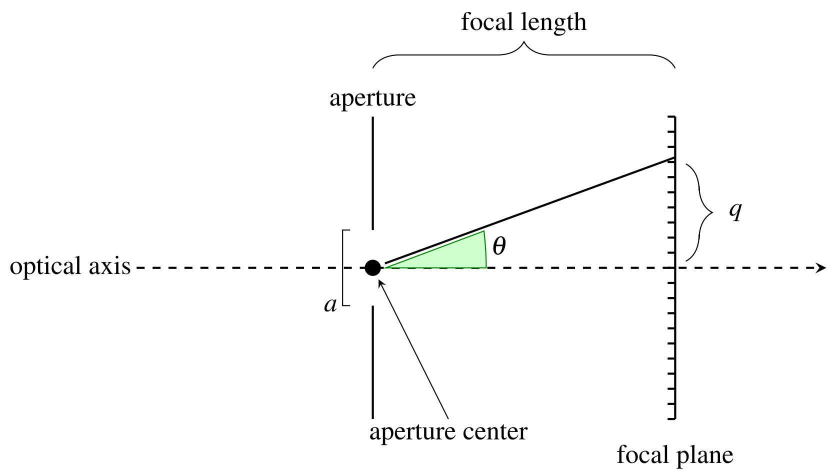


Figure 1: Illustration of the aperture geometry used to calculate the PSF.

To approximate the PSF of a sensor, a Gaussian is specified such that the Gaussian FWHM is the same as the PSF FWHM. This can be accomplished by setting Equation 16 equal to the FWHM of the PSF and solving for σ.

$$\sigma = \frac{1.028\lambda N}{2\sqrt{2\ln 2}} \tag{17}$$

Figure 2 shows the Gaussian approximation overlaid on the PSF.

2. Background Subtraction

An observation from the sensor consists of four elements: the sensor's PSF, the target contribution, the background contribution, and sensor noise. These are related by

$$z = h \otimes (T(x,y) + B) + \omega \tag{18}$$

where z is sensor output, h denotes the PSF, $\otimes$ is the convolution operator, $T(x,y)$ is the target model, B is a static function of the background scene intensity, and $\omega \sim \mathcal{N}(0,r)$ is independent and identically distributed (IID) for each pixel.

Using Equation 18, the contribution of the target and background are combined in the final PSF-convolved scene. Since the background is assumed to be static, the estimation problem is simplified by removing it from the scene. Using the distributive property of convolution, Equation 18 is rewritten

$$z = h \otimes f(x) + h \otimes B + \omega \tag{19}$$

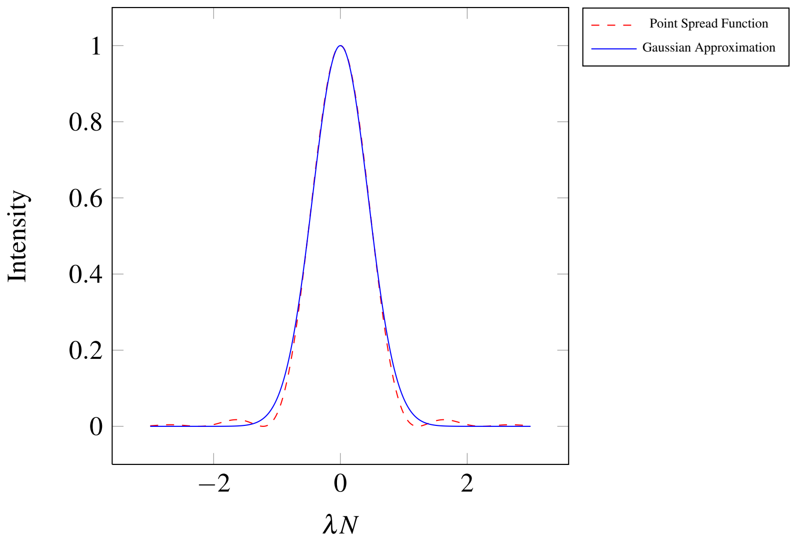


Figure 2: Comparison of Gaussian and PSF.

Equation 19 separates the background contribution from the target contribution without the need to deconvolve the image first. This makes it possible to remove the background contribution from the scene using only estimates of the background determined from the observation frames.

As discussed in Chapter II, a number of methods exist for performing background subtraction. Many of these methods require a large data set of background images in order to build a model. While a good background modeling choice improves the signal-to-clutter ratio (SCR) of a scene, a bad model can remove pixels necessary for target estimation. Based on the results of experiments described in Chapter IV, we chose to use the median filter method of background subtraction due to its simplicity, high speed, and relatively high F-score. The background subtracted observation is denoted

$$\bar{z} = z - \hat{B} \tag{20}$$

3. Sensor Likelihood

We formulate the likelihood measure using $\bar{z}$. Due to the PSF, the target signal will be spread into neighboring pixels. This signal can be used to estimate the subpixel location of the target by comparing the expected intensity values of a pixel and its neighbors with the observed pixel values in a frame. To estimate the intensity values of a central pixel and its neighbors, we generate a template using the ensquared energy of the Gaussian approximation to the PSF.

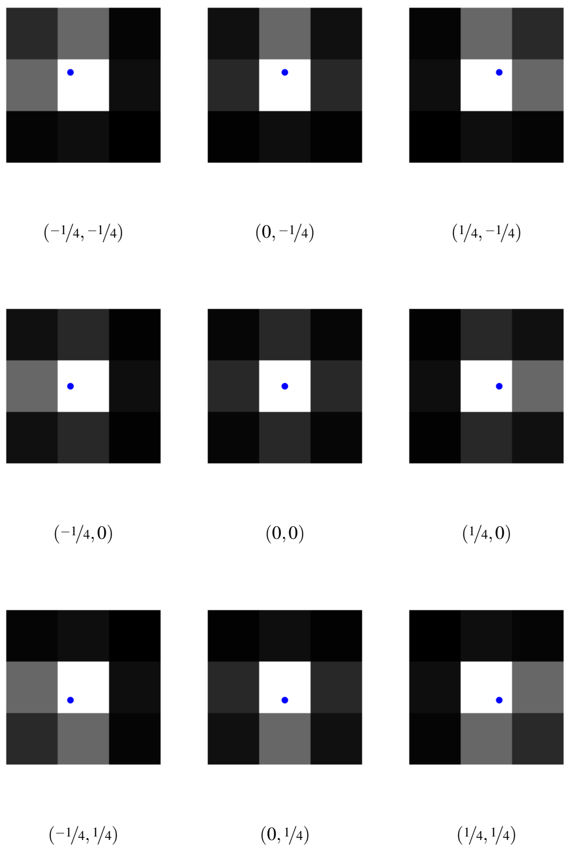


Figure 3: Templates generated by a subpixel target at $(-1/4, -1/4), \ldots, (1/4, 1/4)$. The dot indicates the actual target position for each template.

For square pixels with extents of $[-0.5, 0.5)$, the ensquared energy of a pixel is the fraction of the volume of the Gaussian sampled by an individual pixel and the total Gaussian volume.

$$EE(x,y,\sigma) = \frac{\int_{-0.5}^{0.5}\int_{-0.5}^{0.5} \exp\left(-\frac{(i-x)^2+(j-y)^2}{2\sigma^2}\right) \mathrm{d}i\mathrm{d}j}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right) \mathrm{d}i\mathrm{d}j} \tag{21}$$

The denominator of Equation 21 is the inverse of the normalizing constant for a bivariate Gaussian distribution. Since the pixel noise is IID, this can be further simplified.

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right) \mathrm{d}i\mathrm{d}j = \frac{2\pi}{\sigma} \tag{22}$$

The numerator of Equation 21, however, does not have an exact solution. This is calculated using an adaptive Simpson quadrature technique as implemented in Matlab® 2011.

For a critically sampled sensor with a target in the center of a pixel, a 3×3 template will sample 99.92% of the total target energy. As the target moves to a corner of the center pixel (e.g., $(-0.5, -0.5)$), the template contains a total of 98.16% of the total target energy. Since the target energy is very close to the sensor noise at low signal-to-noise ratio (SNR) values, a larger template size does not increase the amount of target information enough to justify the increased computational cost of a larger template. Figure 3 shows the ensquared energy of the PSF over a 3×3 pixel template for a number of subpixel positions.

The likelihood function is calculated by generating a set of templates with the target located in a discrete set of locations. Discrete positions are calculated from a uniform grid within a single pixel. A spacing parameter, ρ, determines how many subdivisions to use. The discrete subpixel locations are the set of points $(i, j) \in \varepsilon \times \varepsilon$, where $\varepsilon = \{-1/2, -1/2 + 1/\rho, -1/2 + 2/\rho, \ldots, -1/2 + \rho-1/\rho\}$. For example, if $\rho = 5$, a total of 25 templates are generated for target locations $(x, y) \in p \times p$, with points $p = \{-0.5, -0.3, -0.1, 0.1, 0.3\}$.

The likelihood of a subpixel target located at a given position within a pixel is calculated using a template matched filter. The matched filter is an optimal method for calculating the correlation between a finite template and an infinite input [82]. In the case of additive white Gaussian noise (AWGN), the matched filter in the two dimensional discrete case is the sum of squared error between the template, $\mathscr{T}$, and a window centered at a given pixel location, $\bar{z}^{(x,y)}$. The size of the window, N, is the same as the template size.

$$\mathscr{T} \circ \bar{z}^{(x,y)} \triangleq \sum_{i=-N}^{N}\sum_{j=-N}^{N} \left(\mathscr{T}(u,v) - \bar{z}(x+i, y+j)\right)^2 \tag{23}$$

Given a set of templates, S, for each template, $\mathscr{T} \in S$, the correlation is calculated over each pixel in $\bar{z}$. Since $\bar{z}$ is corrupted by noise, ω, Equation 23 is rewritten as the least squares estimator using matrix notation. With a slight abuse of notation, $\mathscr{T}$ and $\bar{z}^{(x,y)}$ are used to represent column vector representations of the corresponding 3×3 matrix used in previous equations.

$$\widehat{\mathscr{T} \circ \bar{z}^{(x,y)}} \triangleq \mathscr{T}^T R^{-1} \bar{z}^{(x,y)} \tag{24}$$

Due to changes in scene lighting, camera white balance settings, shadows, and other changes in scene intensity, the template intensity and observation intensity will not match. This leads to a problem where a template is an exact match for a window centered at pixel (x, y) up to a proportional constant. Furthermore, since the intensity of the target in unknown, the template does not match the target intensity either. Samson et al. address this problem by calculating the maximum likelihood estimate of the target intensity for any given window [50].

For any given template, $\mathscr{T}$, the actual target contribution is $\alpha\mathscr{T}$ and the noise contribution is $\omega \sim \mathscr{N}(0, R)$ (from Equation 18). From [50] the maximum likelihood estimate for α is

$$\hat{\alpha} = \frac{\mathscr{T}^T R^{-1} z^{(x,y)}}{\mathscr{T}^T R^{-1} \mathscr{T}} \tag{25}$$

Combining with Equation 24, the likelihood equation becomes

$$\ell\left(\bar{z}^{(x,y)} | X = \left[\begin{array}{cccc} i & \dot{i} & j & \dot{j} \end{array}\right]^T\right) = \frac{\left|(\mathscr{T}^{(i,j)})^T R^{-1} (\bar{z}^{(x,y)})\right|^2}{(\mathscr{T}^{(i,j)})^T R^{-1} \mathscr{T}^{(i,j)}} \tag{26}$$

The full state is specified for notational convenience. The target velocity is not used in the likelihood function; therefore, the likelihood of a target position is independent of the velocity.

Since R is a diagonal matrix of IID noise r, Equation 26 can be rewritten as:

$$\ell\left(\bar{z}^{(x,y)} | X = \left[\begin{array}{cccc} i & \dot{i} & j & \dot{j} \end{array}\right]^T\right) = \frac{r\left|\mathscr{T}^{(i,j)} \circ \bar{z}^{(x,y)}\right|^2}{\mathscr{T}^{(i,j)} \circ \mathscr{T}^{(i,j)}} \tag{27}$$

Using the correlation operator, $\circ$, in Equation 27 allows the likelihood to be calculated using existing, highly optimized code for cross-correlation operations. While normalized cross-correlation is more commonly used in template matched filtering, the normalization occurs for the template and image combination and does not apply to multiple templates. To allow for direct comparison of results between multiple templates and the image, only standard cross-correlation is used to calculate the likelihood.

C. MOTION MODELS

The PMV algorithm works with both linear and non-linear motion models with additive Gaussian noise. To test the performance of PMV, we define the nearly constant velocity (NCV) model from Bar-Shalom [5]. The following provides a brief description of this model.

The NCV model assumes that a target is moving at a fixed velocity with noisy changes in velocity. The state for this model consists of position and velocity in the x and y dimensions. To avoid ambiguity between position and an instance of the random variable X, the state values are denoted using i and j instead.

$$X = \begin{bmatrix} i & \Delta i & j & \Delta j \end{bmatrix}^T \tag{28}$$

The state update equation is given as:

$$x_k = Ax_{k-1} + \eta \tag{29}$$

With a linear transition matrix

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{30}$$

The noise, η, is additive Gaussian with covariance matrix Q.

$$\eta \sim \mathcal{N}(0, Q) \tag{31}$$

$$Q = \begin{bmatrix} \frac{q}{3}\Delta k^3 & \frac{q}{2}\Delta k^2 & 0 & 0 \\ \frac{q}{2}\Delta k^2 & q\Delta k & 0 & 0 \\ 0 & 0 & \frac{q}{3}\Delta k^3 & \frac{q}{2}\Delta k^2 \\ 0 & 0 & \frac{q}{2}\Delta k^2 & q\Delta k \end{bmatrix} \tag{32}$$

Since η is a Gaussian, the motion probability for this model is:

$$p(X_t|X_{t-1}) = (2\pi)^{-2}|Q|^{-\frac{1}{2}} \exp^{-\frac{1}{2}(X_k - AX_{k-1})^T Q^{-1} (X_k - AX_{k-1})} \tag{33}$$

While the NCV model is a linear model, it is also possible to use the Viterbi algorithm with nonlinear motion models provided they exhibit additive Gaussian noise. The methods explained in Section 2 explain how a linear or nonlinear motion model can be used.

D. SOLVING FOR MAXIMUM A POSTERIORI PROBABILITIES

Given a hidden Markov model (HMM) representing the distribution of target state and observations over time, it is possible to calculate the most likely state given a set of observations and previous states (maximum likelihood (ML) estimation), or to calculate the most likely sequence of states to produce a given sequence of observations (MAP). The PMV algorithm performs the latter calculation. In Bayesian theory, the most likely sequence of states is calculated from the full joint probability, $p(Z_{1\ldots t}, X_{0\ldots t})$. For brevity, the subscript notation $X_{n\ldots m}$ is used to indicate the sequence $X_n, X_{n+1}, \ldots, X_m$.

$$p\left(Z_{1\ldots t}, X_{0\ldots t}\right) = p\left(X_0\right) \prod_{k=1}^{t} p(Z_k|X_k) p(X_k|X_{k-1}) \tag{34}$$

If the realized values for random variables $Z_{1\ldots t}$ are known, and the distribution for X_0 is known, then the inference problem requires solving for the values of $X_{1\ldots t}$ that maximize Equation 34.

$$\mathrm{MAP}(X_{1\ldots t}) = \operatorname*{argmax}_{X_{0\ldots t}} p(X_0) \prod_{k=1}^{t} p(Z_k|X_k) p(X_k|X_{k-1}) \tag{35}$$

The argmax operator can be moved inside the product by noting that the product in Equation 35 is only maximized when $p(Z_k|X_k)p(X_k|X_{k-1})$ is maximized for each X_k. Thus

$$\mathrm{MAP}(X_{1\ldots t}) = p(X_0) \prod_{k=1}^{t} \operatorname*{argmax}_{X_k} p(Z_k|X_k) p(X_k|X_{k-1}) \tag{36}$$

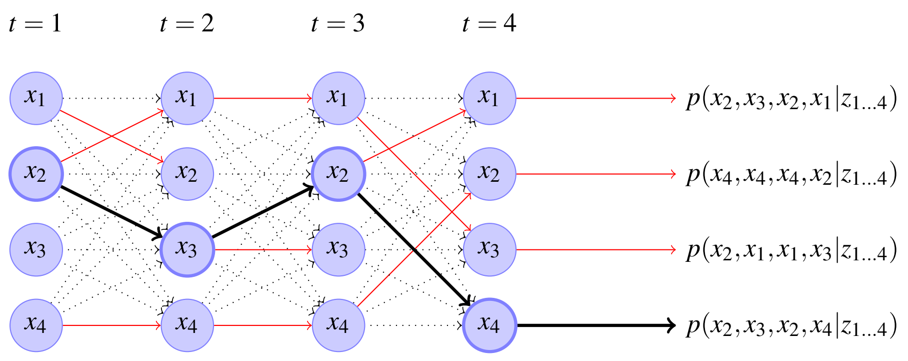
Equation 36 exhibits the two criteria necessary to use dynamic programming methods: optimal substructure and overlapping sub problems [32]. If the random variable X_k is discrete, then the solution to Equation 36 can be calculated in time $O(t \cdot |X|^2)$ using the Viterbi algorithm (the $|\cdot|$ operator in this case refers to the cardinality of the domain for the random variable X) [32].

1. The Viterbi Algorithm

Using sensor model likelihood in Equation 27 and motion probability in Equation 33, the MAP is calculated as

$$\hat{X}_{1\ldots t} = p(X_0) \prod_{k=1}^{t} \operatorname*{argmax}_{X_k} \ell(\bar{z}_k|X_k) p(X_k|X_{k-1}) \tag{37}$$

The naïve implementation of this calculation requires $O(n^t)$ time where $n = \rho^2 wh$, when each frame has size $w \times h$. This implementation is computationally intensive for more than a few frames. Since the number of possible states has been discretized, the Viterbi algorithm can be used to optimally calculate the MAP in $O(tn^2)$ time instead [32].



$t = 1$
$t = 2$
$t = 3$
$t = 4$
x_1
x_2
x_3
x_4
$p(x_2, x_3, x_2, x_1 | z_{1...4})$
$p(x_4, x_4, x_4, x_2 | z_{1...4})$
$p(x_2, x_1, x_1, x_3 | z_{1...4})$
$p(x_2, x_3, x_2, x_4 | z_{1...4})$

Figure 4: Example of the Viterbi Algorithm for a 4 state lattice over 4 time steps. The solid red arrows represent the maximum prior probability for each state. The thicker solid black arrow represents the sequence of states resulting the final MAP estimate.

The Viterbi algorithm is a dynamic programming method that calculates the optimal path through an acyclic directional graph given node and edge costs [33]. The HMM representing each possible discrete target state can be viewed as an acyclic directional graph where each state at time t connects to each state at time $t+1$. The node value for each state is calculated using Equation 27, and edge weights from t to $t+1$ are calculated using Equation 33. The Viterbi algorithm calculates the MAP of the states by selecting the maximum inbound edge for each successive set of states. The states' weights are then updated by multiplying the weight of the inbound edge with the current state likelihood. Outbound edges are calculated as the product of the motion model probability and the node weight. The maximum probability node in the last time sequence represents the final node of the MAP sequence. A simple backtracking step follows the maximum inbound edge for each node, providing the final sequence of states in the MAP. Figure 4 shows an example of the Viterbi algorithm for a simple 4 state lattice over 4 time steps.

The state used in the motion model consists of both position and velocity. The sensor observation, on the other hand, only provides an estimate of the target's position. This means that the velocity must be determined from the position estimates. For the NCV model, the new velocity is calculated from $x_k$and x_{k-1}. Since discrete positions are used for the state, the possible velocity values are also discrete. In this case, the current velocity for a given x_k is simply calculated from the prior x_{k-1}.

For non-linear motion models with additive Gaussian noise, the parameter space is sampled uniformly to build a histogram of possible transitions. The maximum bin in the histogram is the maximum mode of the distribution, and the additive Gaussian noise results in a Gaussian distribution with mean at the location represented by the maximum bin. The distance transform (DT) algorithm discussed in Section 2 can be used for this motion model.

2. Distance Transform Optimization

Assuming the noise in Equation 33 is additive Gaussian noise, the DT algorithm presented in Felzenswalb and Huttenlocher [83] reduces the computation time from $O(tn^2)$ to $O(tn)$. The DT algorithm calculates the maximum or minimum manifold of a set of parabolic or conic functions over a given domain. The set of functions consists of tuples (h,k), where $f(x) = a(x-h)^2 + k$. The variable a determines the size and direction of the parabola. To apply the DT algorithm, the value of a must be constant over all parabolas in the set.

The DT algorithm only works in one dimension, however, since Q is a Gaussian covariance matrix, it is separable (i.e., it can be expressed as the outer product of two vectors, v and h), and thus it is possible to run DT once for each dimension of the state—in this case the x and y axis of the target position.

The distance transform algorithm calculates the maximum or minimum manifold of a set of parabolic or conic functions over a given domain. The set of functions consists of tuples (h,k), where each tuple represents a parabola of the form

$$f(x) = a(x-h)^2 + k \tag{38}$$

The variable a determines the size and direction of the parabola. To apply the distance transform algorithm, the value of a must be constant across all parabolas in the set.

The following is a proof that distance transform calculates the maximum manifold of $p(\hat{X}_{k-1})p(X_k|X_{k-1})$. This proof assumes that Q is separable. Based on this assumption, we only consider the first dimension of $p(\hat{X}_{k-1})p(X_k|X_{k-1})$ without loss of generality. This dimension has the distribution $\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right)$, with $[\ \mu_1 \quad \mu_2 \quad \mu_3 \quad \mu_4\ ]^T = Ax_{k-1}$, and $\sigma_1 = Q_{11}$.

Theorem D.1. *$p(\hat{X}_{k-1})p(X_k|X_{k-1})$ can be expressed as a set of parabolas $\{(h_1,k_1),(h_2,k_2),\ldots,(h_n,k_n)\}$ with common a*

Proof. Taking the log yields:

$$\log\left(p(\hat{X})\right) + \log\left(\frac{1}{\sqrt{2\pi\sigma_1^2}}\right) - \frac{(x-\mu_1)^2}{2\sigma_1^2}$$

Setting the values for a, h, and k results in Equation 38:

$$\begin{aligned}
a &= \frac{-1}{2\sigma_1^2} \\
h &= \mu_1 \\
k &= \log\left(p(\hat{X})\right) + \log\left(\frac{1}{\sqrt{2\pi\sigma_1^2}}\right) \\
f(x) &= \frac{-1}{2\sigma_1^2}(x-\mu_1)^2 + \log(p(\hat{X})) + \log\left(\frac{1}{\sqrt{2\pi\sigma_1^2}}\right)
\end{aligned}$$

Since the Q_{11} is constant for each state transition, the value $a = \frac{-1}{2\sigma_1^2}$ is constant for all parabolas in the dimension, satisfying the requirement for DT

$f(x)$ is in the form of the parabolic equation; therefore, $p(\hat{X}_{k-1})p(X_k|X_{k-1})$ can be expressed as a set of parabolas

Felzenswalb and Huttenlocher show that DT calculates the maximum manifold of a set of parabolas parameterized as $\{(h_1,k_1),(h_2,k_2),\ldots,(h_n,k_n)\}$ with common a

The log operator is monotonic; therefore, the maximum over the set of $f(x_k)$ is the same as the maximum over $p(\hat{X}_{k-1})p(X_k|X_{k-1})$. $\square$

E. CONCLUSION

This chapter provides the methodology developed to track subpixel targets with critically sampled optical sensors. Using the sensor PSF target templates are generated over a discrete set of points. The likelihood function is developed using cross-correlation and maximum likelihood methods. A motion model is defined using the NCV model. We describe the use of the Viterbi algorithm to calculate the MAP of the target path. Finally, the distance transform algorithm is applied to the motion model as an optimization to the general Viterbi algorithm. Chapter IV describes experiments designed to test this method against synthetic and real-world data sets.

IV. DESIGN OF EXPERIMENTS

This chapter describes the experiments performed to validate the following claims regarding the pixel-matched Viterbi (PMV) algorithm:

- Estimated paths are better with respect to root mean squared (RMS) error than track before detect (TBD) methods
- The distance transform algorithm results in $O(n)$ time with no decrease in solution quality as number of sensor pixels increase
- PMV exhibits a higher tracker detection rate (TDR) than TBD

In designing experiments to validate the claims made regarding PMV, we first defined metrics to compare performance between PMV and TBD. Next, we tested background subtraction to determine which method provides the best balance between computational load and accuracy. Using the chosen method, we then determined the affect of the tuning parameter ρ on estimation accuracy versus running time. Using the best value of ρ, we tested PMV and TBD against three different sets of imagery data. The methods used to generate each data set are explained in Section D. Estimation error for PMV and TBD for each data set are reported in Chapter V.

A. MEASUREMENTS

Results from experiments are reported using a modified form of metrics from Bashir [78]. Since the PMV algorithm does not attempt to make a detection decision, it is assumed that a target always exists in the imagery under test. Based on this assumption, true negative (TN) will always be 0. Likewise, values for true positive (TP) and false negative (FN) must be measured in terms of a bounding box around each ground truth state rather than as a detection measurement.

Bashir uses a bounding box concept to compute TP and FN [78]. An estimate is counted as a true positive any time the estimated location of the object lies within a box that fully encloses the target. For subpixel objects, the bounding box can be determined by the Rayleigh criteria as $1.22\lambda N$. For a sensor with $r_c = 2.44$, this means that the bounding box will be 2.44 pixels on the side. Using this measure, the upper left corner of the bounding box is $X_k - (1.22, 1.22)$, and the width and height of the bounding box are 2.44 and 2.44 respectively. Any estimate that falls within this box is counted as a TP while estimates outside of the box are counted as FNs.

Using this definition of a bounding box we calculate the detection rate for a single path as the ratio of TPs to total frames, total number of framess (TFs).

$$\text{Tracker Detection Rate} = \frac{tp}{tf} \tag{39}$$

In addition to point-wise statistics, the total path error is used to describe how far the estimated path is from the actual path. The standard measurement of error is RMS.

$$\text{RMS} = \sqrt{\frac{1}{t}\sum_{k=1}^{t}(\hat{X}_k - X_k)^2} \tag{40}$$

While Equation 40 gives an easy to compare value for path error, it can be easily corrupted by large outliers and constant estimation bias. To eliminate outliers and correct for any constant bias, we also report the RMS error calculated for only the TP estimates, RMS—TP. This measure quantifies the quality of the estimator when it has converged on the true target. An estimator with a higher RMS—TP is providing a lower error estimate when the TP rate is equal. A high detection rate, coupled with a low RMS—TP, indicates better quality estimation than a low RMS with low detection rate.

Constant estimation bias refers to paths that follow the ground truth direction relatively accurately, but are offset by some amount. These paths will result in very low detection rate and very high RMS despite tracking the general trend of the path accurately. To determine when this situation occurs, we calculated RMS—LR for each path. The RMS—LR measure determines the Δx, and Δy offsets that result in the least RMS error.

$$\text{RMS—LR} = \arg\min_{\Delta x, \Delta y} \sqrt{\frac{1}{t}\sum_{k=1}^{t}(\text{dist}(\hat{X}_k - (\Delta x, \Delta y), X_k))^2} \tag{41}$$

If the value of RMS—LR is lower than the value of RMS, then the estimator exhibited a constant bias in the estimate. If the value of RMS—LR is equal to the value of RMS, then the estimator is unbiased.

B. BACKGROUND SUBTRACTION

Background subtraction is a necessary pre-processing step for both TBD and PMV. As described in Chapter II, many approaches have been developed for this problem, however, none were developed with subpixel targets in mind. Methods such as mixture of Gaussians (MoG) or Eigenbackgrounds have a high probability of classifying the tails of the target's point spread function (PSF) convolved projection as background. Additionally, different methods' performance regarding local background movement is highly variable. Due to these challenges, we performed experiments on the median, approximate median, and mixture of Gaussians methods to determine which allows for the best overall estimation performance.

This experiment used synthetic imagery containing a known subpixel target and a background. To generate the background scene, we use the background subtraction data set developed by Brutzer et al. [84]. This data set provides synthetically generated scenery for each of the background subtraction challenges outlined in Chapter II. Specifically, we focused our tests on the data sets for camouflage, darkening, and night.

The quality of various background subtraction methods has been measured in a number of different ways. We are interested in ensuring that the method chosen eliminates as much of the background as possible while eliminating as few of the pixels associated with a subpixel target as possible. This means ensuring the method chosen not only classifies the central pixel containing the target as foreground, but that it also classifies that pixel's immediate neighbors as foreground, as those neighbors contain valuable information on the target position via the PSF. To ensure this criteria, we scored each method for precision and recall, placing a high weight to the importance of precision.

To calculate precision and recall, we counted the number of TPs, false positives (FPs), and FNs for each test image. A pixel is counted as a TP if the method identified it as background and it was actual background. A FP is counted for each pixel that is identified as background but is a pixel containing a subpixel target, or one of the 8-connected neighbors of that pixel. A FN occurs when a pixel that is known background is not classified as background.

To score the methods, we used a F-score criteria. We first calculate precision as

$$p = \frac{tp}{tp+fp} \tag{42}$$

and recall as

$$r = \frac{tp}{tp+fn} \tag{43}$$

and finally the F-score as

$$F = 2\frac{pr}{p+r} \tag{44}$$

Using this method, the median background subtraction method was determined to have the best overall desirable characteristics, and this method was used for all further tests.

1. Data Set Modifications

Since the data sets provided by Brutzer et al. [84] do not contain subpixel targets, we created a modified set for testing. Using the ground truth background images, a 7x7 circular target was inserted in each image using a constant velocity motion with $x_0 = \begin{bmatrix} 792 & -2.74 & 586 & -3.51 \end{bmatrix}^T$. The intensity of the target was set to mean image intensity for each frame. This allows the target to fluctuate intensity with the overall scene. The images with targets where then subsampled with each new pixel calculated as the mean of each 8×8 distinct window in the original image, resulting in a 100×75 pixel image. This subsampled image was then convolved with a Gaussian PSF with full width at half-maximum (FWHM)$= 1/2\sqrt{2\ln 2}$.

Using the frames with targets, we generated two sets of data, one with static background, and one with dynamic background. The original ground truth data consists of a street scene with buildings, roads, traffic lights, and a tree. The tree is considered background, and its leaves blow in the breeze. The dynamic background data set uses the full ground truth frame to include the moving tree. The static background data set uses the dynamic data set with the tree cropped out of the frames. The crop rectangle for the static data set has upper left corner of (53,0) and lower right corner of (100,75).

C. PIXEL DISCRETIZATION

The PMV algorithm discretizes pixels into $\rho \times \rho$ subpixel locations. The quality of the solution and computation time are both affected by the value of this discretization. To determine the affect of ρ on the characteristics of the algorithm, we conducted a series of tests over varying values of ρ. The test data consists of synthetic imagery generated at 20, 10, and 5dB signal-to-noise ratios (SNRs) as described in Section 1. The value of ρ is expected to have the same effect regardless of SNR value.

For each observation sequence, the tracker detection rate, RMS and RMS—TP were calculated, as well as the total computation time for each sequence. The expected result is that increases in the value ρ result in increased computation time and asymptotically decreasing RMS. The tracker detection rate should not be affected by changes in ρ. Section B provides results of these tests.

D. DATA GENERATION

To systematically test the quality of PMV on subpixel targets, we developed three different data sets. The first data set, consisting of synthetically generated image sequences, was used to test the algorithm over a series of SNRs ranging from $1dB$ to $20dB$ with known target states. The synthetic data set does not simulate a background. The second set of data, quasi real-world data, uses real-world imagery of larger than subpixel objects. This data is then processed to convert the targets to subpixel size. The final data set consists of real-world imagery of a known subpixel-sized target in a scene.The following sections describe how each data set was generated.

1. Synthetic Data Generation

Synthetic data was generated for three different image sizes, 30×30, 85×85, and 200×200. For each image size, sets were generated with SNR ratios ranging from $1dB$ to $20dB$. The same ground truth

	Synthetic	Quasi Real-World	Real-World
Simulated Noise	X		
Known Ground Truth	X	X	
Static Background		X	X
Dynamic Background		X	X
Non-simulated			X

Table 1: Parameter values used for synthetic imagery generation.

target motion was used for all generated data sets, regardless of size or SNR. To account for the affects of stochastic noise, a total of 20 individual sequences were generated for each combination of image size and SNR. For all images, the PSF is approximated using a Gaussian with $\sigma = \frac{1.028}{\sqrt{2\ln 2}}$, which approximates the PSF of a perfectly circular aperture and assumes that the pixel size is set to $\frac{1}{r_c} = 0.4098\lambda N$. The value for λN is set to 1 for convenience.

To generate a synthetic image sequence, the target signature is generated on a blank image using the ensquared energy (Equation 21). This simulates a Gaussian-convolved target signal located in a known location with no noise. Next, image noise is calculated for the desired SNR using the maximum target intensity value after convolving with the PSF. This calculates the noise for peak SNR over the image sequence, guaranteeing that the SNR will never exceed the specified value, but due to intensity distribution over multiple pixels, the frame-level SNR can be less.

$$\sigma = \max(EE(X_{gt}))10^{\frac{-\text{snr}}{20}} \tag{45}$$

Using the calculated value for σ, for each pixel in the image we sample from $\mathcal{N}(0, \sigma^2)$ and add the result to the pixel's intensity.

The ground truth used for all of the synthetic images is linear, starting at position (2.0, 2.0) and moving with velocity of (0.5,0.2) for a total of 54 time steps with $\Delta t = 1$. Target intensity is set to $\alpha = 1$.

2. Quasi Real-world Data

The quasi real-world data consists of two data sets captured with an infrared (IR) camera. In both data sets, the captured targets are larger than subpixel. To simulate subpixel targets while still comparing to target ground truth, the maximum dimension of the target was determined by manual inspection. Given a maximum dimension, d, the image size is reduced by half by calculating the average intensity of each group of 2×2 pixels. The resulting image is halved again using this method. We perform a total of n halvings, such that $\frac{d}{2^n} \leq 1$. This produces a lower pixel count image that is equivalent to capturing an image using a sensor with a pixel pitch exactly 2^n larger than the actual sensor pixel pitch.

The size reduction of the images removes the PSF of the sensor. In order to use the quasi real-world images, we convolve the final image size with a Gaussian filter with $\sigma = \frac{1.028}{2\sqrt{2\ln 2}}$. This results in an image with real-world target signature and background, but a simulated PSF. These sequences are used to test background subtraction methods on a reasonably challenging background. Additionally, since the target is large enough in the full size images, it's center of mass can be determined manually, providing ground truth data for the targets.

Two quasi real-world data sets are used, one provided by the Ohio State University IR database (data set 05, plane motion) [85], referred to as the "plane" data set, the other captured locally using a FLIR SC 8200 mid-wave infra-red (MWIR) camera of a city street using a 50mm lens at f/4 aperture setting, referred to as the "car" data set. The plane data set consists of an aircraft flying away from the camera from above a cloud bank into the clouds. The clouds in this set are moving throughout the sequence, and present a dynamic background. This set contains a total of 491 images of size 80×60.

In the car data set, images were captured of cars driving on a straight road. The images were captured from the top of a 4 story building, looking towards the street, which goes up a hill 1.75 miles away. The resulting perspective shows movement up the hill as vertical motion in the sequence. This sequence contains a cluttered, but mostly static background. The car data set contains a total of 301 images of size 32×51. Figure 5 shows three frames from each of these sequences.

3. Real-world Data

Real-world data was captured using a FLIR 8200 MWIR camera with 50mm lens and f/4 aperture setting. According to the manufacturer, the pixel pitch of the sensor is $18\mu m$. The spectral range of the camera is $3-5\mu m$ wavelength frequencies. This results in a PSF with FWHM of

$$\begin{aligned} \text{FWHM} &= 1.028\lambda N \\ &= 1.028 \cdot 4 \cdot 4 \\ &= 16.448\mu m \end{aligned}$$

The full width to first zero ring is

$$\begin{aligned} \text{FW} &= 1.22\lambda N \\ &= 1.22 \cdot 4 \cdot 4 \\ &= 19.51\mu m \end{aligned}$$

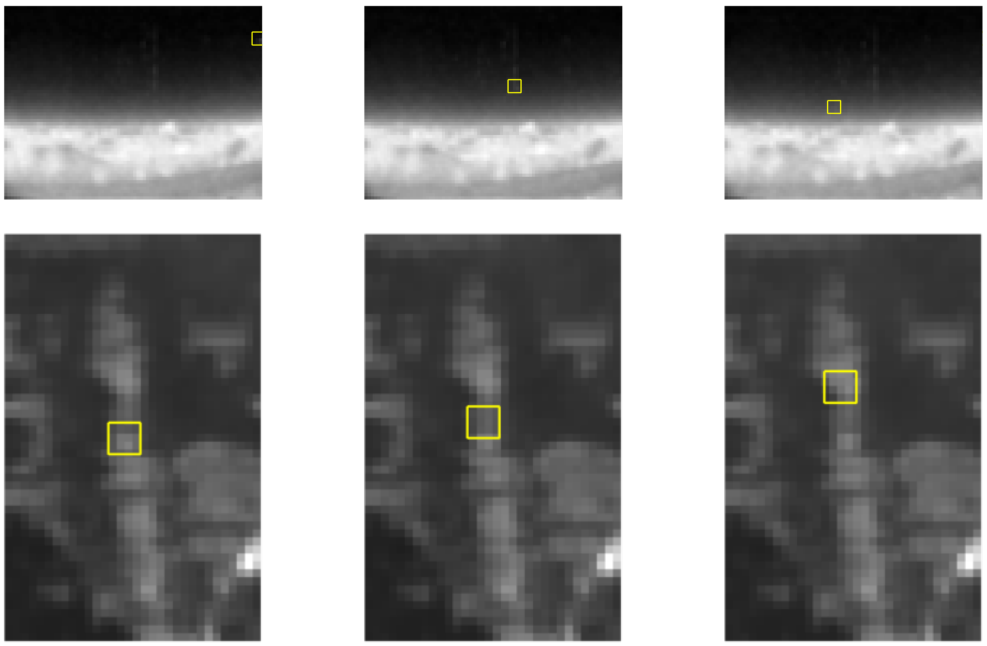


Figure 5: Three frames for the "plane" and "car" quasi real-world data set. A box is drawn over the target in each frame. The top row contains frames 1, 245, and 491 from the plane data set. The second row contains frames 1, 150, and 301 from the car data set. All images have been contrast enhanced for clearer viewing.

Using these numbers, the pixel sampling rate is $^{\mathrm{FW}}/_{\mathrm{pitch}} = 1.36$. This means the camera is undersampled for the PSF size, however, since the full width of the spot is greater than the pixel pitch, the target intensity is still spread to more than one pixel.

For this data set, an IR emitter in the $3\mu m$ wavelength is used as a target. The emitter is a SVF360–8M LED from Cal Sensors, Inc. The 14mm diameter sensor was placed on a vertical holder 97m from the camera. The LED was powered at 1.25V using a constant current circuit, with a 60% duty cycle, 10kHz, pulse width modulation (PWM) signal to control emitter intensity. To simulate linear motion, the emitter's holder was attached to a Pioneer P3DX robot set to move at a constant rate of 0.4m/s.

We attempted to estimate ground truth of the sensor location using a larger heat source on the robot that is offset a constant distance from the emitter. The manually estimated path, however, exhibited higher variability than either the PMV or the TBD estimated path. Rather than use the manual path as a ground truth, we instead compare the paths generated by the two methods to each other. Inspection of the image sequence with the overlaid path shows that both methods provide realistic estimates of the emitter's path.

To avoid adding multiple targets to the image, the final sequence of images used was cropped to remove the robot and heat sink from the test imagery. Figure 6 shows three frames from the test sequence

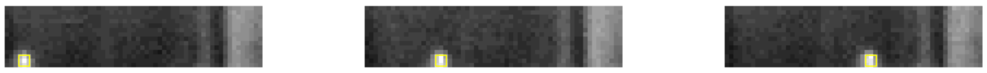


Figure 6: Frames 47, 87, and 126 of the real-world data set. A box is drawn over the target in each frame. All images have been contrast enhanced for clearer viewing. The emitter is located ~12 pixels above the bottom edge of the image.

with the heat sink included, and three frames of the cropped sequence containing only the emitter. The real-world image sequence consists of 126 frames of 59×14 sized images.

E. IMAGE SIZE VS. COMPUTATION TIME

The size of an image is expected to impact the quality of the TBD method while having little to no impact on the quality of the PMV method. On the other hand, TBD is expected to show nearly constant run time over different image sizes, while PMV is expected to show linear run time. In the case of TBD, larger images result in less particle coverage of the possible distribution, reducing the quality of the estimate. Since the PMV method fully samples each pixel in the image, the estimation quality is expected to remain the same regardless of image size. The computational time for each method, on the other hand, is the opposite of the expected estimation quality. As the number of pixels increase, PMV's running time is expected to increase linearly while TBD will remain constant for a given particle population size. To verify these expectations, we report the running time for each of the synthetic image data sets. Timing results are reported for each of the three types of image data set in their respective sections in Chapter V.

V. RESULTS AND DISCUSSION

This chapter provides results and analysis of those results for the experiments described in Chapter IV. The performance of the median, approximate median and mixture of Gaussians (MoG) background subtraction methods are discussed in Section A. Section B analyzes the effects of increased pixel discretization on the quality of the estimated path and the running time of pixel-matched Viterbi (PMV). In Section C, we compare the run-time performance of PMV and track before detect (TBD) for different input sizes. Sections D and E present the mean root mean squared (RMS) performance for each filter on the synthetic and quasi real-world sets, as well as the detection rate for each method over varying signal-to-noise ratio (SNR) values. Finally, Section F compares the paths produced by each method for a real-world target.

A. BACKGROUND SUBTRACTION

Figure 7 shows the precision-recall curves for the median, approximate median, and MoG background subtraction methods. The logarithmic axes are used to emphasize the relationship of these methods, however, the overall precision and recall values for the subpixel target are well below those reported in [53] for larger targets using the same background data. The F-score for each of the three methods is shown in Figure 8. The score for dynamic and static background is given for each method.

Each of the three background subtraction techniques show strengths and weaknesses in the test data. In precision and recall, the approximate median and median methods both provide better results than the MoG method, however, the precision for all three methods fall below 10%, while recall is less than 2%. The poor recall values are due to the low SNR ratio of the target, as well as the small target area. In the test data set, the number of target pixels ranges from 4 to 9 pixels depending on the location of the target. The total image size is 75 by 100. Since background subtraction methods typically segment large foreground objects such as pedestrians and cars, even with a low peak SNR value, the large target area results in much more information to exploit versus the subpixel problem. While the camouflage data set attempts to measure the performance of the background subtraction methods with low SNR targets, the results presented in [53, 52, 54] have a peak SNR of 15.13 with an average area of 11,313 pixels, as opposed to a subpixel target with area ≤ 1 pixels. The poor performance of MoG in this particular case is due to the similarity between target intensity and background intensity. In order to increase recall, a very small threshold value is necessary. This small value negatively impacts the precision of the MoG filter.

Due to dynamic background elements in the data set, a low threshold value results in higher classification of the background motion as foreground instead. This results in the low precision values due to a

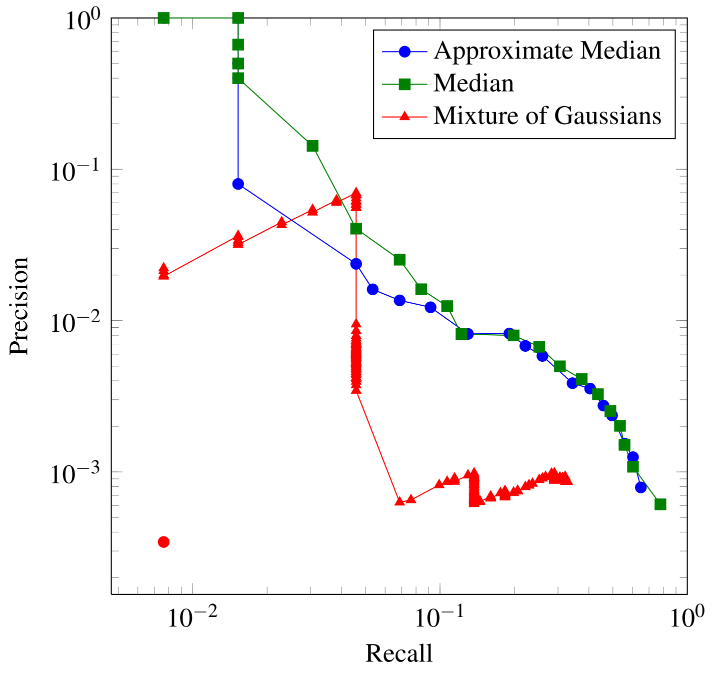


Figure 7: Precision-recall curves for approximate median, median, and mixture of Gaussians background subtraction with a subpixel target in the foreground.

large number of incorrect foreground classifications. The MoG method, however, achieves a higher F-score than the two median methods. This is due to the ability to model each pixel with multiple possible intensity distributions, allowing MoG to better account for the dynamic, repetitive motion. Unfortunately, while MoG provides a better F-score, Figure 7 shows that the improvement is mostly in precision, while recall does not improve noticeably from the static background.

The results indicate that the chosen background subtraction method must take into account the expected dynamics of the background scene. For example, if a large Δt value is used, background can be expected to change in illumination and content to some degree over that interval. This requires a method that can adapt quickly to global scene changes. Likewise, dynamic background components will have a strong effect on the performance of both the background subtraction method and the target estimator. None of the methods tested perform strongly in all of these regimes, but each method has strengths that can be tailored to known scene conditions.

The overall poor precision and recall scores for the tested methods given a subpixel target indicate that the subpixel problem is novel enough to warrant a background subtraction method that is tailored to suit the problem domain. Each of the tested methods were developed for use on multi-pixel targets with reasonable SNR values and large target areas. The subpixel target presents challenges that these methods do not attempt to address, such as low SNR and aperture diffraction effects. Additionally, each of these methods is tested as

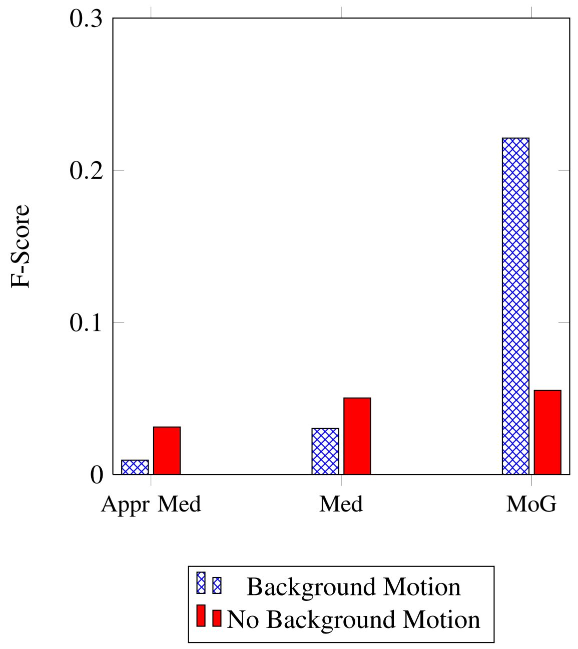


Figure 8: Comparison of F-scores for approximated median, median, and mixture of Gaussians background subtraction with a subpixel target in the foreground.

a stand-alone operation. We did not attempt to modify them to use possible target locations as a conditioning variable in the threshold decision. While not tested in this research, it is possible that the estimation method and background subtraction method can be built to complement each other with feedback, resulting in better background removal as well as better estimation.

Despite the shortcomings of each of the background subtraction methods, results from the quasi real-world data set confirm that background subtraction is a critical component of PMV. For example, the RMS error for PMV on the plane sequence without background subtraction is 25.04. This same sequence using median background subtraction has a RMS error of 0.3808. It is unclear whether the background subtraction method achieves a limit to the amount of improvement possible or if further gains in accuracy can be achieved with better background subtraction techniques. Further research in this area is necessary to quantify the impact of background subtraction for different frame sequences, and to determine whether feedback from the estimator can be leveraged to improve the background subtraction accuracy.

B. PIXEL DISCRETIZATION

Figure 9 shows the RMS for ρ values of 2, 6, 10, 14, and 18 at SNR values of 1, 5, 10, 15, and 20 dB. Figure 10 shows the affect of each ρ value on the total runtime. Values shown are average run times with variance less than 0.025s over 20 tests for each ρ value.

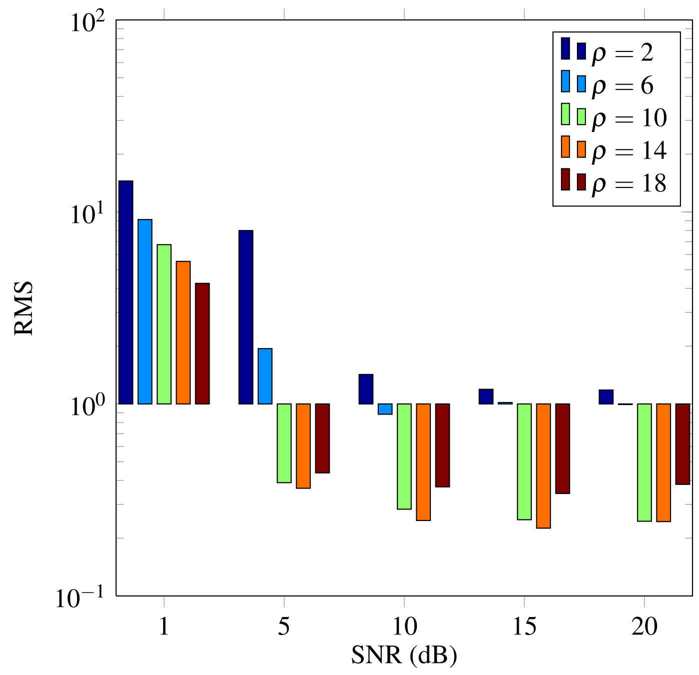


Figure 9: RMS versus SNR for $\rho \in \{2, 6, 10, 14, 18\}$.

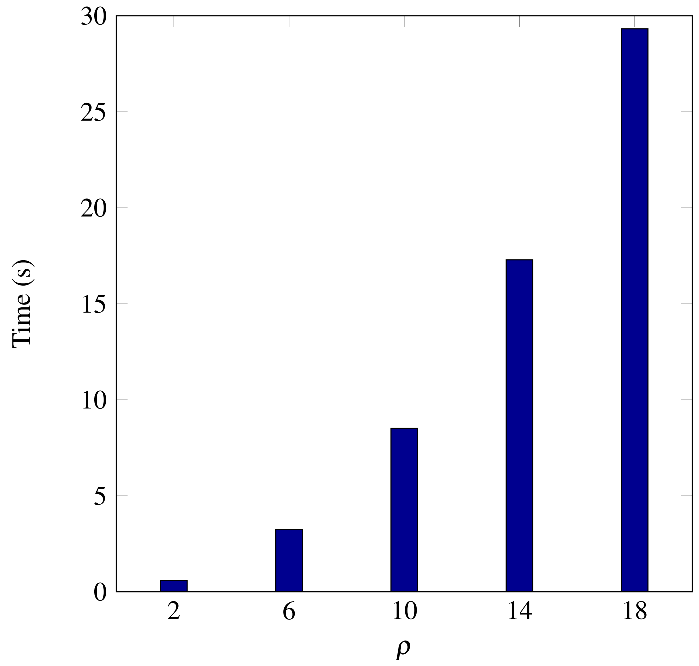


Figure 10: Mean runtime over 20 tests for each value of $\rho \in \{2, 6, 10, 14, 18\}$.

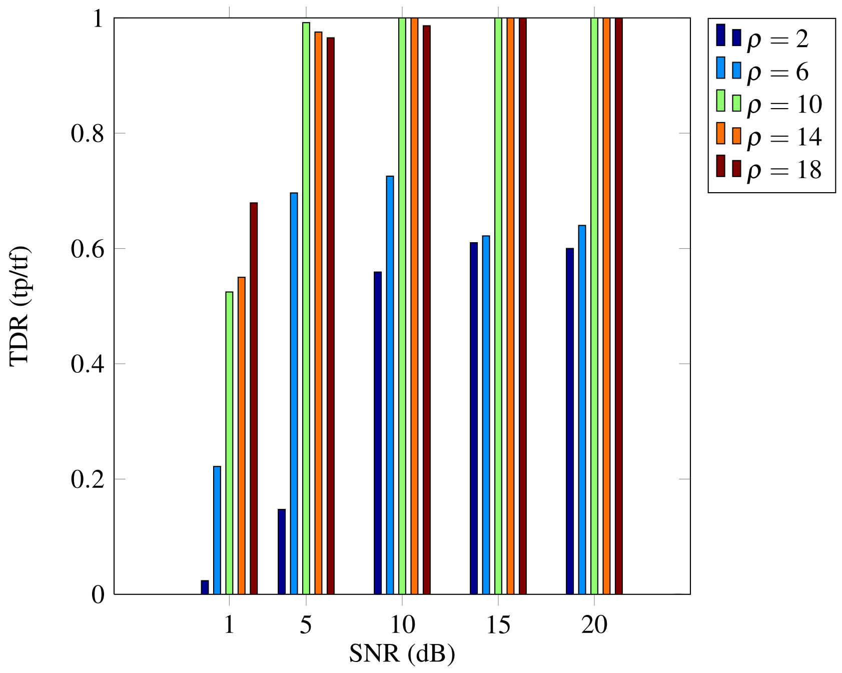


Figure 11: Mean TDR over 20 tests for each value of $\rho = \{2, 6, 10, 14, 18\}$.

The effects of the number of discrete pixel positions used in the discretization of the pixel space are shown in Figures 9 and 10. As ρ increases, the estimation accuracy increases up to a value of $\rho = 14$. At $\rho = 18$, the estimation accuracy decreases again. The increase in accuracy is expected. For small values of ρ, fewer subpixel positions are sampled, and the RMS can only decrease to the size of the discretization bin. As the number of bins increases with higher ρ values, the ability to provide more accurate target locations translates directly to lower RMS. Given this explanation, however, the expected response to increased ρ is for the RMS to asymptotically approach a lower bound imposed by the sensor noise. Instead, when $\rho = 18$, we see a decrease in RMS relative to $\rho = 10$ and $\rho = 14$.

The reason for the discrepancy in RMS for increasing values of ρ is unknown. Further experiments show an additional increase in RMS for $\rho = 22$, followed by a decrease for $\rho = 26$ and $\rho = 30$, although the RMS at $\rho = 30$ is still higher than $\rho = 10$. A possible explanation is that the higher ρ values cause the probability density to be spread across too many points, resulting in Viterbi arbitrarily breaking ties between equal probability states, however, this explanation does not fully account for the increase and subsequent decrease in RMS values, only the increase. Give the low RMS value achieved for all $\rho > 10$, however, this discrepancy is not large enough to warrant further investigation with regard to the best value of ρ to use.

Since increases in ρ result in a ρ^2 increase in the total number of states estimated with the Viterbi algorithm, the computation time is expected to increase exponentially with increasing ρ. Figure 12 shows the average run time of PMV over 20 tests at each ρ value. The increase in run time and decrease in RMS suggest that a possible improvement to the PMV method would be to run it in two passes, using the $\rho = 1$ value to determine the pixels most likely included in the target path, then again with $\rho = 10$, performing estimation only on those pixels and their neighbors selected in the first pass. This iterative improvement method could lead to increased speed with equal accuracy relative to running PMV once with a high ρ value. For very large images, the timing improvement may be significant.

C. COMPUTATION TIME

Table 2 shows the average run time in seconds for each set of synthetic image sizes, as well as the quasi real-world and real-world sequences. Each run was performed on 2.8 GHz quad-core Intel Xeon processor with 24GB of 800 MHz DDR2 RAM with nominal operating system processes running in the background. The same information is shown in Figure 12 with the x-axis set to number of frames times number of pixels. This format helps to emphasize the linear trends of both methods as pixel count increases.

The run times in Table 2 and Figure 12 show the linear time complexity of PMV with respect to number of frames and number of pixels. The time complexity of TBD is also linear with respect to number of particles, however, as seen in Figure 12, the running time is affected by other factors as well. The main reason for the variability in run times for TBD is in the jump Markov model used by the filter. As the number of particles with existence state set to true increases, the number of motion model evaluations increases. This results in two different complexity models with different constant values associated with them, however, both are linear with respect to the number of particles used. The car data set results in a large increase in computation time because TBD is attempting to track a number of targets simultaneously with greater than 90% of the population in the existence model for the majority of the run. A similar characteristic can be seen in the increase in running time for the real-world data set. In both cases, greater than 90% of the particle population is in the existence model, while the other data sets typically average at 75% of the population in this model. While this means scenes that result in high probability of detection for TBD will require more computation time, the overall complexity for TBD is still linear with respect to the number of particles, and lower than PMV.

D. SYNTHETIC IMAGERY

Figure 13 shows the mean RMS over 20 runs of synthetic data for each image size. The horizontal dotted line indicates the RMS at which estimation achieves subpixel accuracy. Figure 14 shows the average RMS estimate for only those points that fall within the ground truth bounding box region for each ground truth position for both TBD and PMV on the three synthetic data sets. Figure 15 shows the average tracker

		Image Size	# Frames	Mean Run Time (s)	Std of Run Time (s)
PMV	Synthetic	30x30	55	17.99	2.49
		85x85	55	58.38	1.78
		200x200	55	403.77	8.34
	Plane	80x60	491	422.49	-
	Car	32x51	301	76.92	-
	Real-World	50x41	106	34.67	-
TBD	Synthetic	30x30	55	1.30	0.05
		85x85	55	1.94	0.07
		200x200	55	6.39	0.61
	Plane	80x60	491	15.42	-
	Car	32x51	301	39.60	-
	Real-World	50x41	106	3.57	-

Table 2: Run times for PMV and TBD for synthetic, quasi real-world, and real-world data sets. Mean run times are reported for synthetic imagery over 20 separate tests, quasi real-world and real-world report actual run time for a single instance.

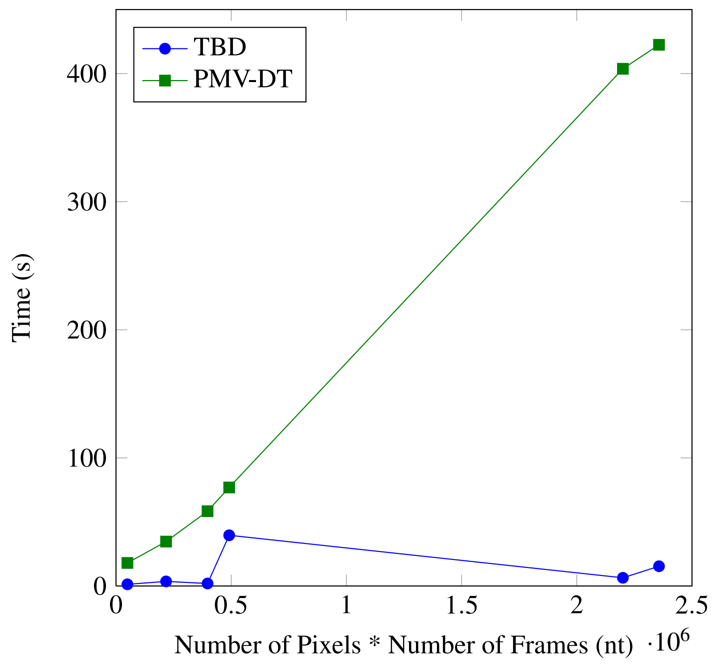


Figure 12: Run times for TBD and PMV for synthetic, quasi real-world, and real-world data sets. Mean run times are reported for synthetic imagery over 20 separate tests, quasi real-world and real-world report actual run time for a single instance .

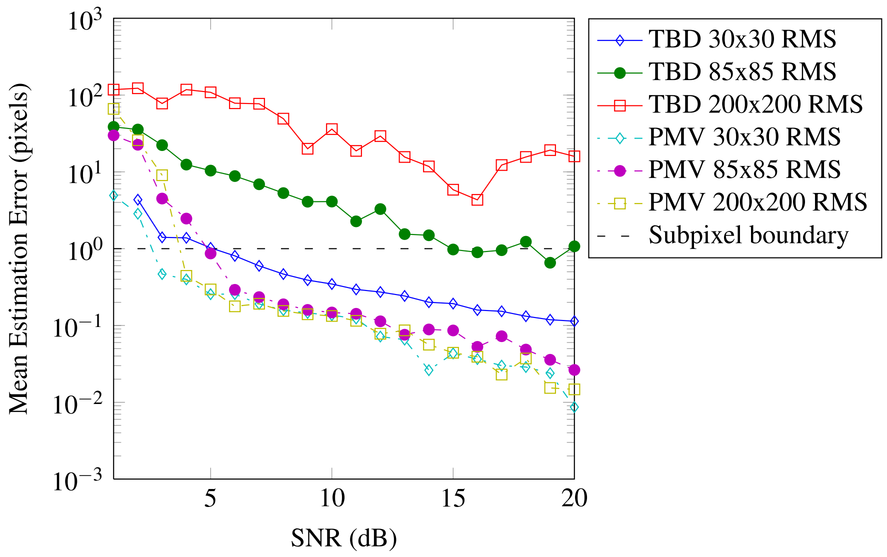


Figure 13: Mean RMS over 20 test runs from 1dB to 20dB SNR for PMV and TBD.

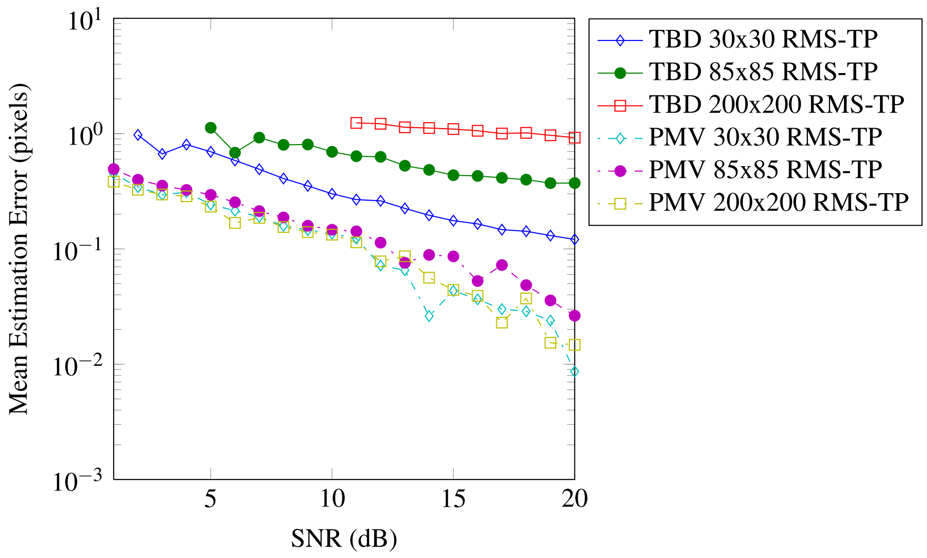


Figure 14: Mean RMS—TP over 20 test runs from 1dB to 20dB SNR.

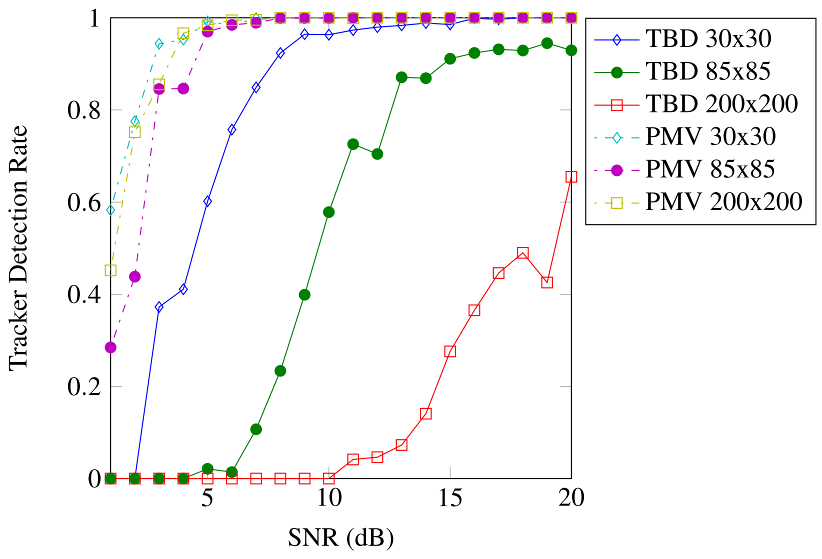


Figure 15: Mean TP over 20 test runs from 1dB to 20dB SNR.

detection rate for TBD and PMV for each of the three synthetic data sets based on the percentage of estimated points that fall within the bounding box of a ground truth point.

Using the RMS measure of performance, PMV outperforms TBD in every synthetic data set for SNRs greater than or equal to 4dB. At SNR values from 1 to 4dB, the methods are relatively comparable, however, both methods fail to achieve subpixel estimation error. The tracker detection rate for PMV is greater than 80% for SNR of 3dB and greater, with TBD producing greater than 80% true positive rate starting at 7dB for 30x30 images, and 12dB for 85x85. TBD fails to achieve greater than 80% detection rate for the 200x200 data set at any tested SNR. Using the RMS—TP measure, PMV outperforms TBD on all synthetic data sets.

The image size has relatively little impact on the quality of the PMV solution. For all three image sizes, PMV achieves better than 80% tracker detection rate at SNR of 3dB or greater. As shown in Figures 13 and 15, the RMS and detection rate are closely clustered across image sizes. In relation to TBD, which is image size dependent, PMV does not appear to reduce quality with image size. However, as expected, the linear growth of PMV with respect to the number of pixels results in much longer computation time.

The computational load of PMV can be partially addressed through parallelization techniques. Profiling shows that the distance transform (DT) method accounts for a total of 37% of the total run-time. The manner in which this method iterates over each possible motion parabola in the set of possible motions makes it difficult to parallelize, however, it is possible to perform parallel instances of DT on spatially segmented portions of the image using a MapReduce paradigm [86]. The Map function consists of mapping spatially segmented sets of parabolas to processing units in order to solve the DT for those units. The Reduce function calculates the minimum of the mapped results. This requires that each segment overlaps it's neighboring segments by the maximum expected target velocity. If target velocity cannot be limited in practice, the parallelization may not be possible, however, for many targets, the maximum velocity is limited by sensor frame rate and target kinematics.

The RMS—TP shown in Figure 14 shows the best performance of each method when all outliers that are not within a 1 pixel bounding box of the target location are excluded. The TBD methods do not extend all the way to 1dB SNR due to poor tracking performance resulting in no true positive values to plot. When only using true positive values, PMV outperforms TBD for all synthetic data sets. This shows that, even under ideal circumstances, the variance of the TBD method is greater than the PMV method. The most likely reason for this is that TBD relies on a population of particle hypothesis that are adjusted at each time step. Each adjustment due to the motion model injects variation to the TBD estimate. This limits the lower bound performance of TBD. PMV, on the other hand, selects the mode of the motion model at each step. While noise can disturb this estimate, the sequence of states serves to damp out noise spikes. Additionally, the entire maximum log-likelihood sequence is stored for each time step. This means that poor estimates made early in the sequence can be discarded for sequence estimates that result in better log-likelihood in later observations. With no variance injection in the estimate and memory of previous estimates that may have resulted in good solutions, PMV is able to produce lower variance and more consistent estimates than TBD.

Appendix B shows the estimated paths for each of the synthetic image sizes and SNR combinations. In the TBD examples, the paths tend to start in the middle of the image before converging to the target path. The reason for this is that TBD begins to detect the target before it can develop a position estimate. Since the position estimate is based on the location of each particle in the existence set, the population is often dispersed throughout the image when the detection threshold is first met. This causes large initial errors in the path estimation, however, the RMS—TP shows that even when the estimator has converged with the target path, the estimation error continues to be larger than PMV. Increasing the detection threshold to force convergence on the target before registering target detection caused TBD to fail to detect the target at lower SNR values. Figures 21 and 20 show the results of varying the detection threshold for TBD. All results shown here use a threshold of 0.7 which achieved the maximum tracker detection rate (TDR) and minimum RMS across the most SNR values.

The results for RMS—LR, discussed in Section A are not presented. In every instance, the RMS—LR values were the same as RMS. This indicates that neither estimation method has a consistent path-wise estimation bias.

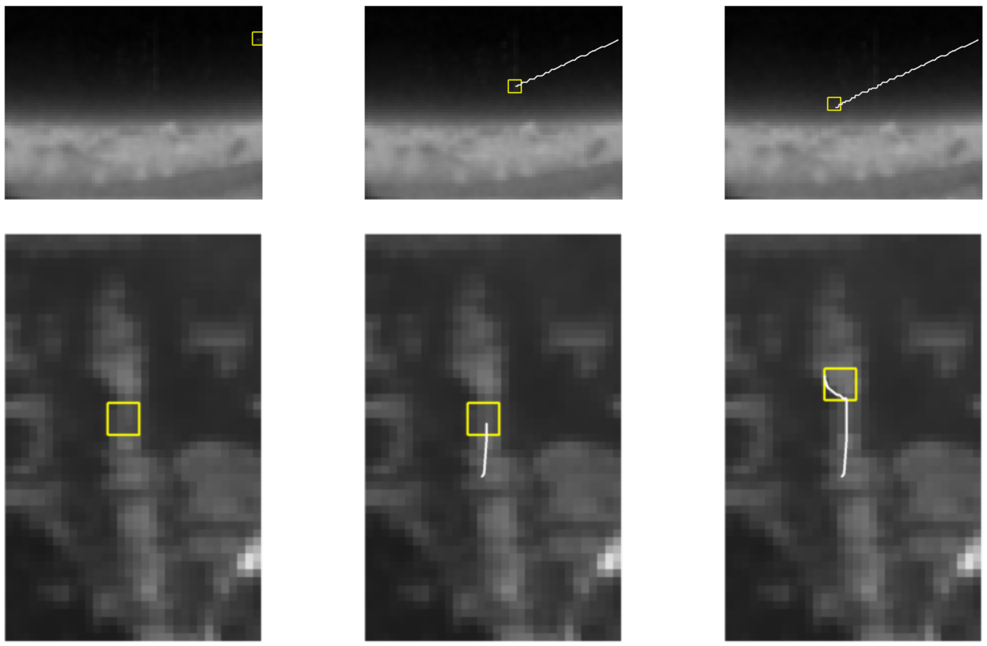


Figure 16: Three frames showing the PMV estimated paths for the "plane" and "car" quasi real-world data set. A box is drawn over the target in each frame, and the current path estimate is drawn as a white line. The top row contains frames 1, 245, and 491 from the plane data set. The second row contains frames 1, 150, and 301 from the car data set. All images have been contrast enhanced for clearer viewing.

E. QUASI REAL-WORLD

The PMV RMS for the car data set is 1.89, and the plane data set is 0.36 with detection rate of 0.66 and 1.0 respectively. RMS for the car data set and the plane data set cannot be calculated for the TBD filter because it did not surpass the detection threshold of 0.7 in either case. Further reductions in the threshold down to 0.5 did not improve this result, while thresholds below 0.5 resulted in large clusters of particles throughout the image with cluster centers mostly tracking background motion in the scene. Figure 16 shows the first, middle and last frames from each data set with a square centered on the actual target position, and the PMV estimate at the current frame. Figure 17 shows the same three frames with the TBD estimated distribution overlaid as red dots on the image state space.

The quasi real-world imagery data set highlights two advantages of PMV over TBD: moving background suppression, and multi-target suppression. The plane data set exhibits a moving background that causes the TBD method to fail, while the car data set contains multiple targets, which violates the single target assumption used for both filters. While PMV properly rejects the moving background components of the plane scene, TBD is unable to converge on the target due to that movement. In the car scene, PMV produces a single track that produces the highest overall likelihood. In this data set, target paths come into

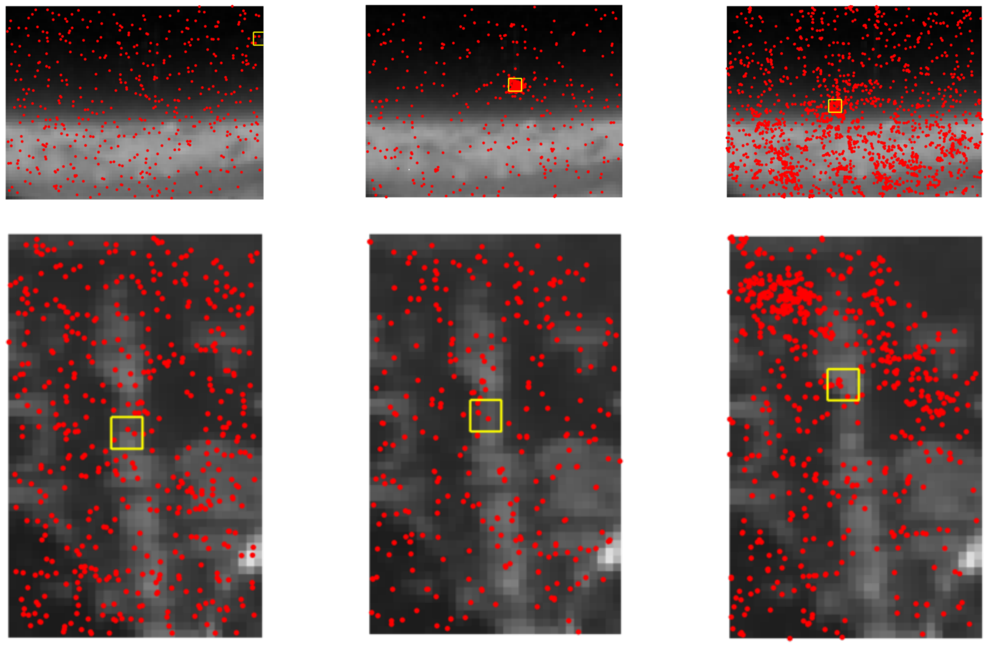


Figure 17: Three frames showing the TBD estimated paths for the "plane" and "car" quasi real-world data set. A box is drawn over the target in each frame, and the current path estimate is drawn as a white line. The top row contains frames 1, 245, and 491 from the plane data set. The second row contains frames 1, 150, and 301 from the car data set. The red dots indicate the particle distribution of the TBD filter at the current frame. All images have been contrast enhanced for clearer viewing.

close proximity, resulting in PMV estimating a path that spans three separate targets. TBD on the other hand spreads particles to each of the targets in the scene, with no target producing a high enough likelihood to result in convergence of the population on a single mode. This produces no path estimate, and the resulting clusters of particles are not clear enough to make any determination of target locations or directions.

In the plane data set, a layer of clouds covers the bottom half of the scene. These clouds are moving slowly, and have a texture that produces high subpixel target likelihoods. Using the median background subtraction filter removes much of the cloud bank from the image, but it does not remove the parts of the clouds that are changing through the scene. Despite this, PMV tracks the plane starting in the first frame of the data set. TBD, on the other hand, converges to the target location within the first five frames, but then disperses across the cloud bank. The remainder of the data set, TBD particles are spread throughout the cloud bank with no particles tracking the airplane. This behavior is due to the maximum likelihood (ML) nature of TBD. When an ML estimate of target location is made in the cloud bank, the majority of particles that were tracking the target are lost in the resampling step. After several time steps, any particles that were tracking the correct target have been lost. In order to reacquire the target, TBD must randomly place a new particle near enough to the target to have a high enough likelihood to be selected more frequently. With high likelihood in the clouds, this does not occur.

Since the PMV method samples every pixel, even if higher transient likelihoods occur over the course of multiple frames, unless those transient likelihoods are very close to the actual target, they do not affect the maximum path that represents the true target motion. Due to this dense estimate, high likelihoods are not "missed" as they can be with TBD. Additionally, since the estimate is based on the maximum prior path for each pixel, unless noise or background motion occurs with higher likelihood very near to the target, the target path is preserved once the noise or motion has ceased to provide high likelihoods. This means that PMV is more easily able to reject temporary motion of background objects.

F. REAL-WORLD

Figure 18 shows the first, middle and last frames from the data set with a square centered on the actual target position, and the PMV estimate at the current frame on the top row. The bottom row of Figure 18 shows the same three frames with the TBD estimated path as a white line.

The real-world data set provides a comparison between PMV and TBD on a known subpixel target. Manual estimation of ground truth data for this set resulted in a highly varying path that is less accurate than both PMV and TBD; therefore, we compare the estimated paths to each other to provide insight into the relative performance of the two methods. Figure 18 shows the estimated paths for the target for various frames. Figure 19 shows a close-up comparison of both methods' estimates. The path estimated by PMV is

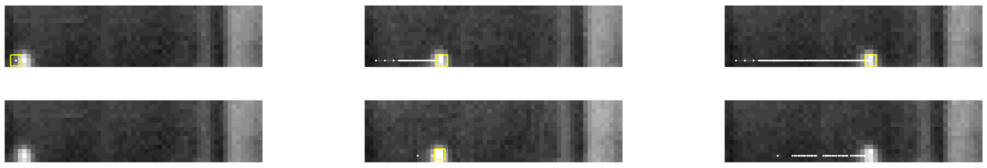


Figure 18: Three frames for the real-world data set. The top row shows the PMV estimated path up to the current frame for frames 47, 87, and 126 with a rectangle placed at the current estimated location. The bottom row shows the TBD estimated path up to the current frame for frames 47, 87, and 126. A rectangle is placed over the current estimate if TBD has passed the detection threshold for the frame (frames 47 and 126 do not have rectangles because TBD has fewer than 70% of particles tracking the target.

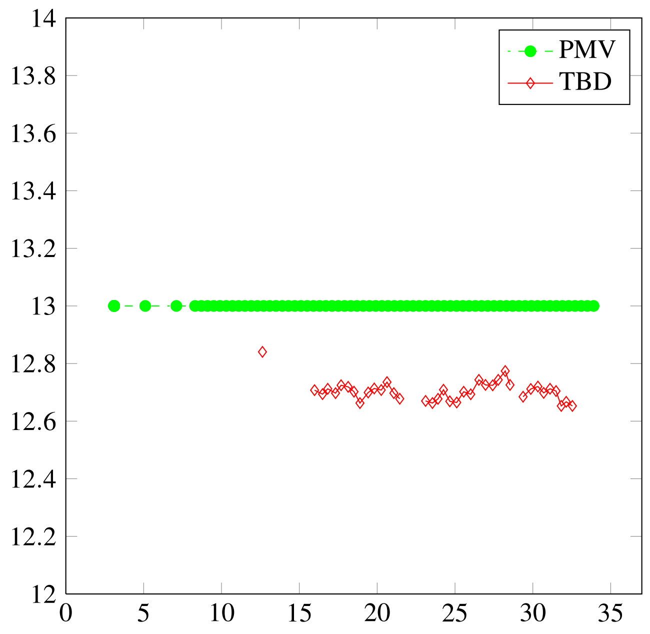


Figure 19: Close-up of the PMV estimated path and TBD estimated path in the real-world data set.

smooth, and follows the correct straight line motion of the emitter. The path estimated by TBD on the other hand has high variance relative to the straight line path. Both paths closely track the emitter's correct position in the data set, however the high variability of TBD indicates that it does not track the target as accurately.

The smoothness of the PMV estimate is due to the use of the entire path in estimation. Low probability estimates at the beginning of the sequence may still be included in the final solution. For TBD, on the other hand, there is no mechanism for keeping track of prior path estimates when the filter has not converged. As seen in the path plots in Appendix B, the estimate tends to be very close to the center of the image until the filter has converged over the course of several frames. As a consequence of this behavior, any estimates made prior to convergence will result in very large error. As discussed in Section D the variability of the TBD population after convergence results in a shifting mean. While the mean stays close to the actual target location, the fluctuation produces poorer results than the maximum a posteriori probability (MAP) result.

While the low noise characteristics of the sensor used to the capture the real-world data set preclude testing at lower SNR values, the observed results are consistent with the predicted results from the synthetic data set. At a peak SNR of 20dB, both methods are expected to produce subpixel position estimates with PMV providing a better estimate. As seen in Figure 19, the paths produced by each method are very close to each other, with the smoother path of the PMV more closely matching the actual motion of the target.

G. CONCLUSION

This chapter presented the results of for the experiments outlined in Chapter IV. Background subtraction precision and recall curves were presented and the median method is shown to provide the best overall performance in the subpixel domain. The RMS error of PMV for varying values of ρ was shown, with $\rho = 10$ chosen as the best compromise between computation time and estimation quality. The computation time of PMV and TBD were compared to show that expected linear growth rate of PMV with respect to the image pixel count and number of frames, while TBD showed an increase in computation time for increasing number of frames. Estimation results for synthetic, quasi real-world, and real-word data sets were shown, with PMV producing lower RMS error and higher TDR than TBD on each of the data sets.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION

This dissertation presents a method for tracking subpixel targets, called pixel-matched Viterbi (PMV). A likelihood function is specified using a template matched filtering and templates generated using the sensor's point spread function (PSF) to determine the likelihood of subpixel targets in a single image. For any motion model featuring additive Gaussian noise, the likelihood and motion model are combined to calculate the maximum a posteriori probability (MAP) of target positions over a given sequence of images using the Viterbi algorithm and distance transform function. The final result consists of the most likely path of the target through the sequence of images.

We compared PMV to a current state of the art tracking method, the track before detect (TBD) particle filter. With respect to mean root mean squared (RMS) error over a series of tests, the estimated path is superior to TBD for images down to 4dB signal-to-noise ratio (SNR) and equivalent to TBD below this level. With regards to number of estimated path points within 1 pixel of the actual target position, the proposed method achieves greater than 80% detection rate with SNR as low as 3dB versus 7dB for TBD. Unlike TBD, the mean RMS is shown to remain consistently low regardless of image size.

Finally, we discussed shortcomings of the presented method such as the need for better background subtraction in the presence of dynamic background motion, and to assume target presence in the image sequence. For backgrounds with many dynamic elements, and image sequences with larger than subpixel objects, the background motion or texture on the larger object causes the tracker to follow the wrong target. While background subtraction partially addresses the dynamic background problem, better methods are needed to further improve the solution. For moving foreground objects that are larger than a pixel, the missed estimation typically occurs due to subpixel texture on the target. Methods to reject or discount larger targets, or to track multiple subpixel targets simultaneously may help alleviate this problem.

The implications of this work can be applied in a number of problem areas such as maritime monitoring and tracking from space, unmanned aerial vehicle (UAV) sense and avoid, stealth aircraft tracking, and many other applications where small targets cannot be easily resolved by an optical sensor. This method allows planners to reduce sensor pixel count without compromising the ability to track certain classes of targets due to size. Additionally, it allows for analysis of low resolution, coarse imagery rather than requiring higher resolution, allowing for bandwidth savings. For applications such as sense and avoid and infrared search and track (IRST) this method has the potential to provide earlier warning for incoming targets than is currently available. This additional warning window allows for more time to take appropriate countermeasures.

A. FUTURE WORK

While the results show that PMV performs better than the TBD method, they also indicate areas for further research. As discussed in Chapter II, Section D the type of scene dictates the best background subtraction method to use. Additionally, the background subtraction methods tested were not developed to use any information from the subpixel domain. An improved method that is tightly integrated with the MAP calculations has the potential to increase the performance of the tracker by eliminating all background-related subpixel textures.

Another area of research that will benefit the PMV method is the ability to perform a detection decision. Currently, PMV assumes that a target is always present in the input data. If no target exists, PMV will return the sequence of states that best describes the noise. While this path is assigned a probability value by the MAP estimator, a better method would determine when a target appears in the data, and when it disappears from the data. Using a detection decision would allow PMV to run on large video data sets with no need to determine whether or not a subpixel target exists in any of the frames.

The performance of PMV precludes its use in real-time imagery. As suggested in Chapter IV, Section C and Chapter V, Section C, a number of techniques may reduce overall processing without reducing estimation quality. For example, adaptive discretization has the potential to reduce the total number of states considered by the Viterbi algorithm. Additionally, parallelization may be employed to increase the frame rate. Through use of parallelization, we believe that PMV can be made to run on live video.

In addition to making improvements on PMV, a number of application areas exist that have the potential to provide useful testing corpora. These will likely introduce scenes with features that were not considered or tested in this work, possibly highlighting strengths and weaknesses not yet identified. One such corpus is the asteroid tracking database collected by the Lowell Observatory as part of their Near-Earth-Object Search (LONEOS). This data set contains thousands of image sequences taken by earth-based telescopes over fixed time intervals. The goal of the LONEOS project is to detect small asteroids in these sequences. Currently this is accomplished through crowd-sourcing methods with individuals looking for minute changes between images. Using PMV, we believe this search can be automated over the LONEOS data set.

Other possible improvements to PMV are to extend it for multiple target detection, and include camera ego-motion in the estimation. These improvements would allow for an expanded set of applications to include aerial target tracking and acquisition. While a number of multi-target methods exist in the literature, none of these methods have been attempted on subpixel targets. These methods are tightly coupled with the detection and estimation steps of the tracking method used. While not trivial, we believe the same techniques can be applied to PMV.

B. CONCLUSION

The primary thesis of this dissertation is that subpixel tracking for optical sensors is possible, and can be accomplished with a similar level of accuracy as current state of the art methods for multipixel targets. Experimental results support this thesis and indicate that even greater accuracy than current state of the art methods is possible. While the proposed method is more computationally intensive that the current state of the art, advances in multicore architectures, vector processing, and embedded processing offer the necessary level of performance to use this method on real-world data. The subpixel tracking method is the first such method the author is aware of in the literature and uniquely fills a gap in capability for the tracking and computer vision communities.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

Figures 20 and 21 show the estimation root mean squared (RMS) error and tracker detection rate (TDR) for the TBD method for varying detection thresholds. The authors of TBD recommend a threshold of 0.6 as the detection decision, however the figures show that, for the subpixel problem set, better results are achieved at the 0.7 thresholds. The value of 0.7 was used for all comparison data runs in Chapter V.

Figures 22, 23, and 24 show the TBD estimated paths for 20 separate observation sequences using the same ground truth motion. The ground truth is overlaid on these in red. These plots show the type of estimation error that can be expected as signal-to-noise ratio (SNR) decreases. Many of the paths begin in the center of the image, then converge to the ground truth path. This is a consequence of the maximum likelihood (ML) estimation technique which produces higher variance estimates in the beginning, with lower variance as more observations are made. The high variance estimates tend towards the center of the image because the filter has exceeded the threshold limit for detection before the variance has reduced enough to produce accurate estimates.

Figures 25, 26, and 27 show the PMV estimated paths for 20 separate observation sequences using the same ground truth motion. The ground truth is overlaid on these in red. Many of the estimated paths down to SNR of 1dB track the ground truth path accurately, however, when PMV fails to track the path, the result is a path that follows a weighted random walk where the motion model weights the walk towards a path that more closely resembles the model's specification.

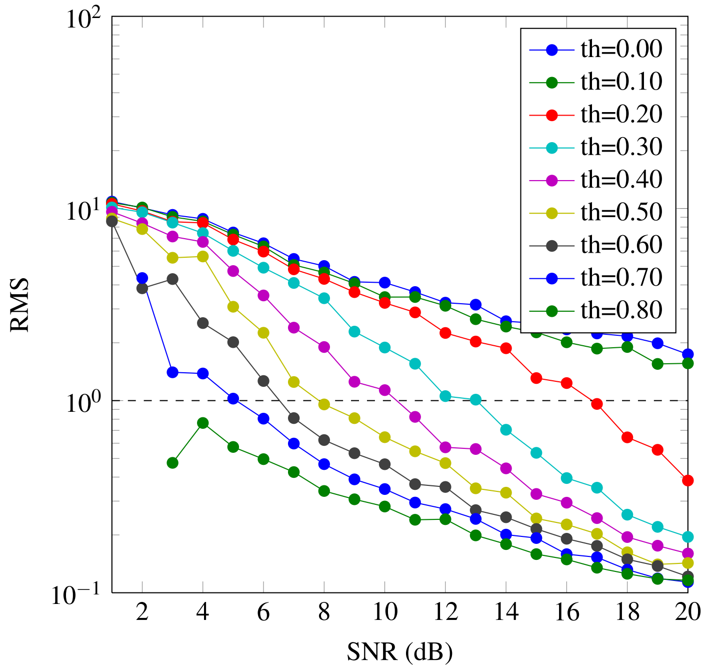


Figure 20: TBD RMS for threshold values from 0 to 1 at 0.1 intervals.

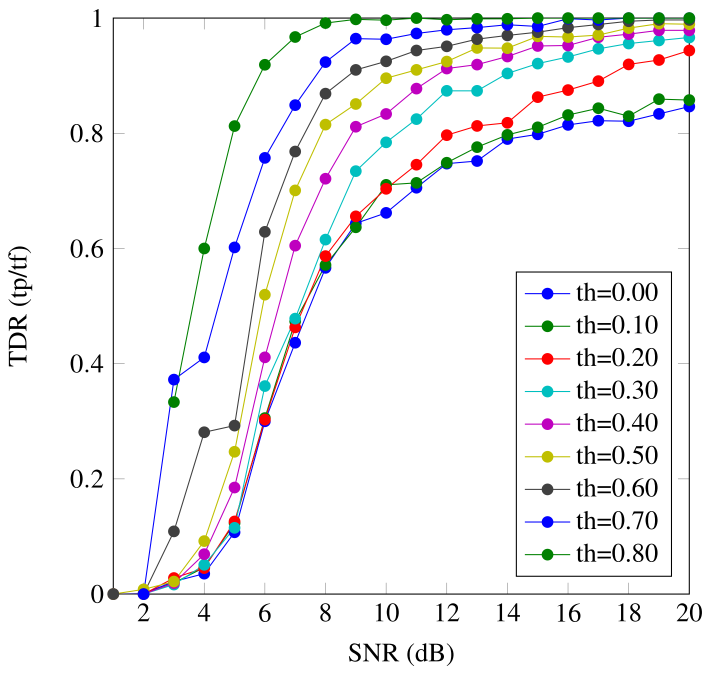


Figure 21: TBD TDR for threshold values from 0 to 1 at 0.1 intervals.

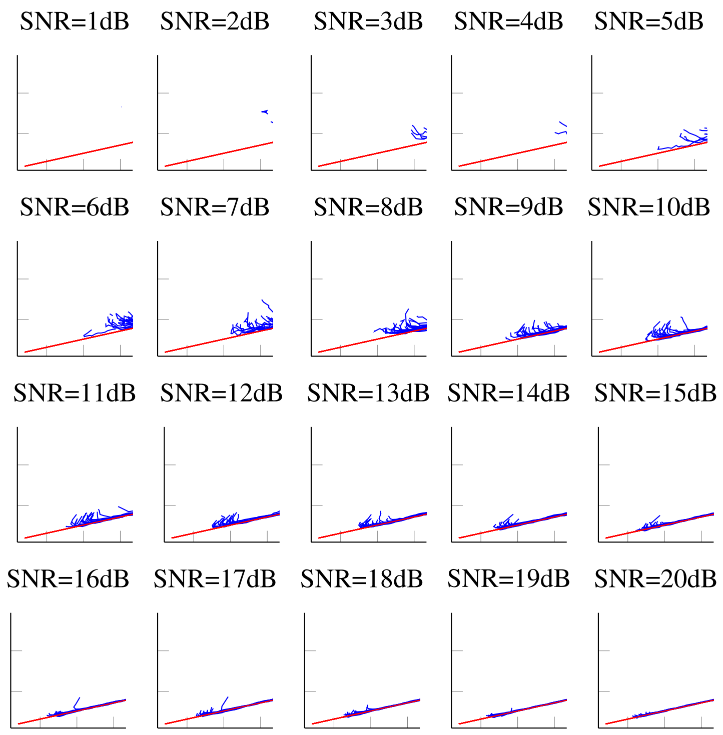


Figure 22: TBD estimated paths for 30×30 images from 1dB to 20dB.

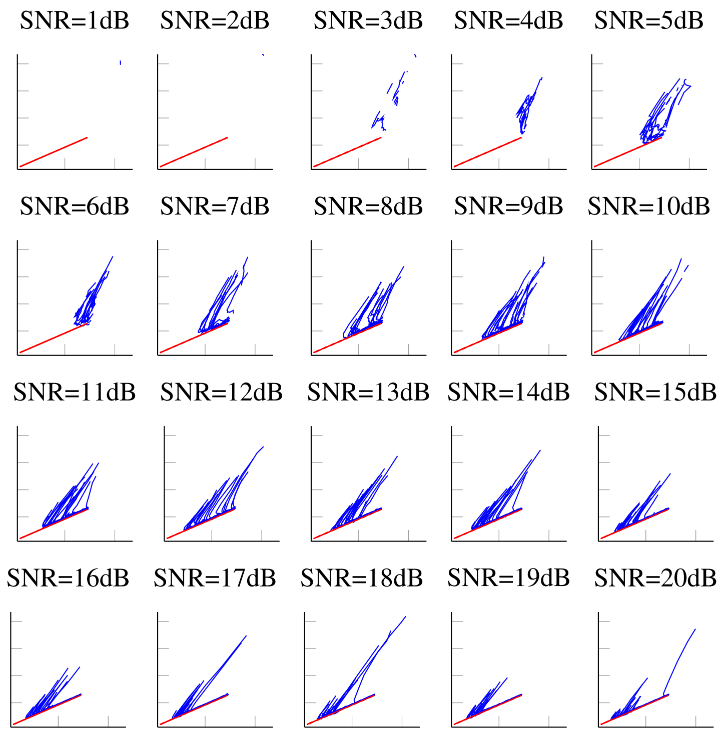


Figure 23: TBD estimated paths for 85×85 images from 1dB to 20dB.

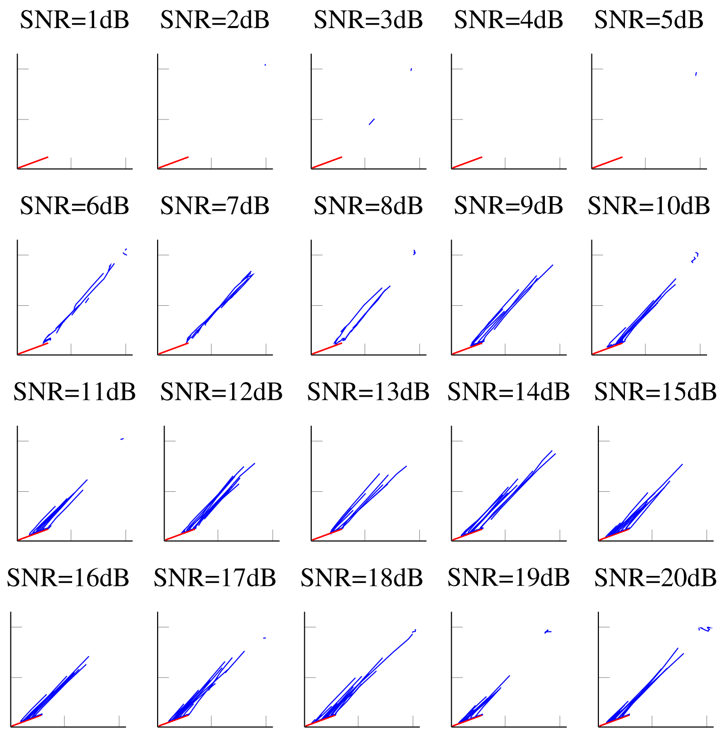


Figure 24: TBD estimated paths for 200×200 images from 1dB to 20dB.

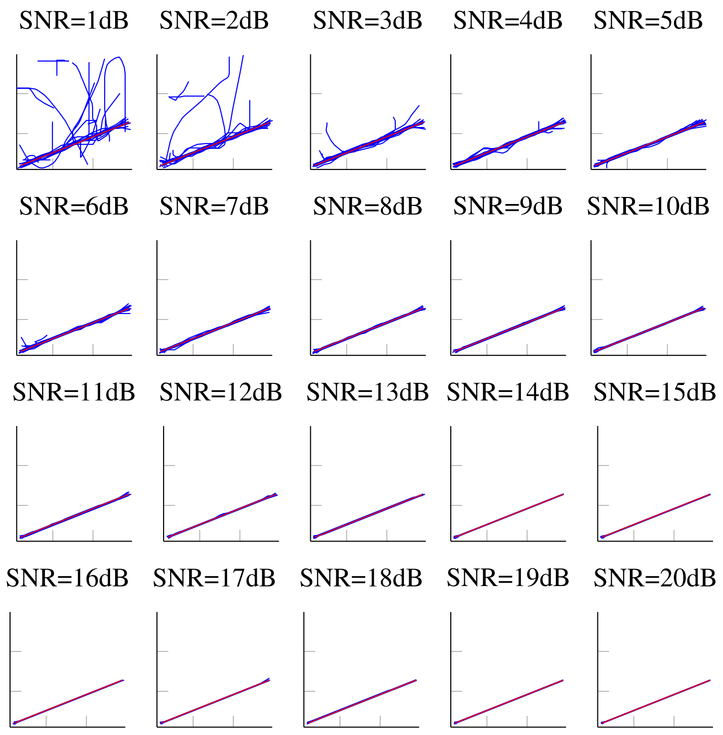


Figure 25: PMV estimated paths for 30×30 images from 1dB to 20dB.

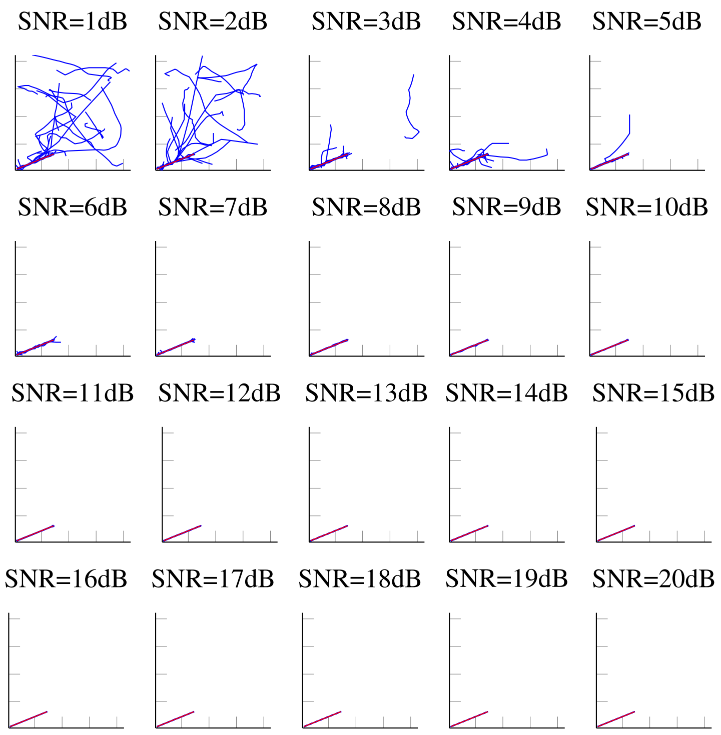


Figure 26: PMV estimated paths for 85×85 images from 1dB to 20dB.

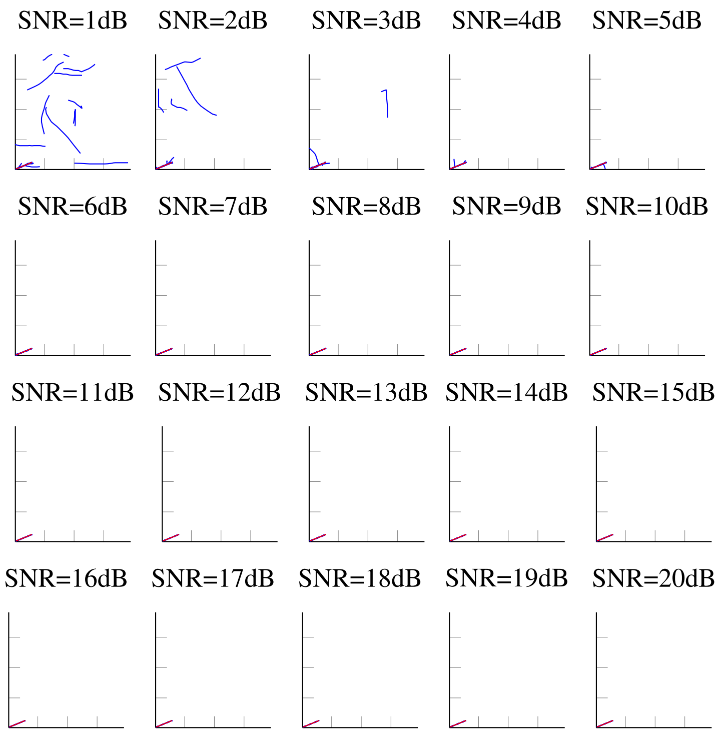


Figure 27: PMV estimated paths for 200×200 images from 1dB to 20dB.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] S. Magnuson, “Military ‘swimming in sensors and drowning in data’,” *National Defense*, Jan. 2010. [Online]. Available: http://www.nationaldefensemagazine.org

[2] E. Lake, “Drone footage overwhelms analysts,” *Washington Times*, Nov. 2010. [Online]. Available: http://www.washingtontimes.com

[3] GeoEye, “Geoeye-1 fact sheet,” GeoEye, Dulles, VA, Tech. Rep., Jan. 2010.

[4] R. Carnie, R. Walker, and P. Corke, “Image processing algorithms for UAV "Sense and Avoid",” in *Proceedings of the 2006 IEE International Conference on Robotics and Automation*. IEEE, May 2006, pp. 2848–2853.

[5] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[6] R. Bhargavi, K. Ganesh, M. Sekar, P. Singh, and V. Vaidehi, “An integrated system of complex event processing and Kalman filter for multiple people tracking in WSN,” in *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, Jun. 2011, pp. 890 –895.

[7] C. Chiang, P. Tseng, and K. Feng, “Hybrid unified Kalman tracking algorithms for heterogeneous wireless location systems,” *Vehicular Technology, IEEE Transactions on*, vol. 61, no. 2, pp. 702 –715, Feb. 2012.

[8] I. Jami, R. Ormondroyd, and E. Artarit, “Improved handset tracking using Kalman filter algorithm aided by angle-spread information from a smart antenna array,” in *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, vol. 2, Oct. 2003, pp. 752 – 756 Vol.2.

[9] S. Julier, “The spherical simplex unscented transformation,” in *American Control Conference, 2003. Proceedings of the 2003*, vol. 3, Jun. 2003, pp. 2430 – 2434 vol.3.

[10] S. Julier, J. Uhlmann, and H. Durrant-Whyte, “A new approach for filtering nonlinear systems,” in *American Control Conference, 1995. Proceedings of the*, vol. 3, Jun. 1995, pp. 1628 –1632 vol.3.

[11] C. Komninakis, C. Fragouli, A. Sayed, and R. Wesel, “Multi-input multi-output fading channel tracking and equalization using Kalman estimation,” *Signal Processing, IEEE Transactions on*, vol. 50, no. 5, pp. 1065 –1076, May 2002.

[12] P. Tseng, C. Chen, and K. Feng, "An unified Kalman tracking technique for wireless location systems," in *Wireless Pervasive Computing, 2007. ISWPC '07. 2nd International Symposium on*, Feb. 2007.

[13] M. Teutsch, W. Kruger, and J. Beyerer, "Fusion of region and point-feature detections for measurement reconstruction in multi-target Kalman tracking," in *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, Jul. 2011, pp. 1 –8.

[14] R. Van der Merwe and E. Wan, "The square-root unscented Kalman filter for state and parameter-estimation," in *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, vol. 6, 2001, pp. 3461 –3464 vol.6.

[15] S. Russell and P. Norvig, *Artificial Intelligence-A Modern Approach*, 3rd ed. Saddle River, NJ: Prentice Hall, 2010.

[16] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor mav," in *International Conference on Intelligent Robots and Systems*, 2012.

[17] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *National Conference on Artificial Intelligence*, pp. 593–598.

[18] Z. Yin and R. Collins, "Belief propagation in a 3d spatio-termporal MRF for moving object detection," in *IEEE Computer Vision and Pattern Recognition(CVPR)*, Jun. 2007.

[19] M. Andriluka, S. Roth, and B. Schiele, "Pictorial structures revisited: People detection and articulated pose estimation," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, Jun. 2009, pp. 1014 –1021.

[20] S. Tonissen and Y. Bar-Shalom, "Maximum likelihood track-before-detect with fluctuating target amplitude," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 34, no. 3, pp. 796 –809, Jul. 1998.

[21] C. Morelli, M. Nicoli, V. Rampa, and U. Spagnolini, "Hidden markov models for radio localization in mixed LOS/NLOS conditions," *Signal Processing, IEEE Transactions on*, vol. 55, no. 4, pp. 1525 –1542, Apr. 2007.

[22] W. Blanding, P. Willett, and Y. Bar-Shalom, "ML-PDA: Advances and a new multitarget approach," *EURASIP J. Adv. Signal Process*, vol. 2008, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1155/2008/260186

[23] Y. Bar-Shalom, F. Daum, and J. Huang, “The probabilistic data association filter,” *Control Systems, IEEE*, vol. 29, no. 6, pp. 82 –100, Dec. 2009.

[24] Y. Junkun, L. Hongwei, W. Xu, and B. Zheng, “A track-before-detect algorithm based on particle smoothing,” in *Radar (Radar), 2011 IEEE CIE International Conference on*, vol. 1, Oct. 2011, pp. 422 –425.

[25] D. M. Greig, B. T. Porteous, and A. H. Seheult, “Exact maximum a posteriori estimation for binary images,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 51, no. 2, pp. pp. 271–279, 1989.

[26] Y. Boykov, O. Veksler, and R. Zabih, “Markov random fields with efficient approximations,” in *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, Jun. 1998, pp. 648 –655.

[27] V. Kolmogorov and R. Zabih, “What energy functions can be minimized via graph cuts,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 65–81, 2004.

[28] C. Fox and G. K. Nicholls, “Exact MAP states and expectations from perfect sampling: Greig, porteous and seheult revisited,” in *Bayesian Inference and Maximum Entropy Methods in Science and Engineering: 20th International Workshop*, vol. 568. Gif-sur-Yvette (France): American Insitute of Physics, Jul. 2000, pp. 252–263.

[29] D. Tran and J. Yuan, “Optimal spatio-temporal path discovery for video event detection,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, Jun. 2011, pp. 3321 –3328.

[30] M. Islam, C.-M. Oh, J. S. Lee, and C.-W. Lee, “Multi-part histogram based visual tracking with maximum of posteriori,” in *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, vol. 6, Apr. 2010, pp. 435–439.

[31] F. Yan, W. Christmas, and J. Kittler, “A maximum a posteriori probability Viterbi data association algorithm for ball tracking in sports video,” in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 1, 2006, pp. 279 –282.

[32] J. Li, “Viterbi algorithm,” 2012, lecture Notes.

[33] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *Information Theory, IEEE Transactions on*, vol. 13, no. 2, pp. 260 –269, Apr. 1967.

[34] ——, “Convolutional codes and their performance in communication systems,” *Communication Technology, IEEE Transactions on*, vol. 19, no. 5, pp. 751 –772, Oct. 1971.

[35] D. Haccoun and G. Begin, “High-rate punctured convolutional codes for Viterbi and sequential decoding,” *Communications, IEEE Transactions on*, vol. 37, no. 11, pp. 1113 –1125, Nov. 1989.

[36] P. Robertson, E. Villebrun, and P. Hoeher, “A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain,” in *Communications, 1995. ICC ’95 Seattle, ’Gateway to Globalization’, 1995 IEEE International Conference on*, vol. 2, Jun. 1995, pp. 1009 –1013 vol.2.

[37] H. Ardö, K. Åström, and R. Berthilsson, “Real time Viterbi optimization of hidden Markov models for multi target tracking,” in *In IEEE Workshop on Motion and Video Computing*, 2007.

[38] D. Klein and C. D. Manning, “A* parsing: Fast exact Viterbi parse selection,” Stanford InfoLab, Technical Report 2002-16, Mar. 2002. [Online]. Available: http://ilpubs.stanford.edu:8090/532/

[39] J. Nelson and H. Roufarshbaf, “A tree search approach to target tracking in clutter,” in *Information Fusion, 2009. FUSION ’09. 12th International Conference on*, Jul. 2009, pp. 834 –841.

[40] J. Feldman, I. Abou-Faycal, and M. Frigo, “A fast maximum-likelihood decoder for convolutional codes,” in *Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th*, vol. 1, 2002, pp. 371 – 375 vol.1.

[41] M. Zeng, J. Li, and Z. Peng, “The design of top-hat morphological filter and application to infrared target detection,” *Infrared Physics and Technology*, vol. 48, no. 1, pp. 67 – 76, 2006.

[42] N. Acito, G. Corsini, M. Diani, and G. Pennucci, “Comparative analysis of clutter removal techniques over experimental IR images,” *Optical Engineering*, vol. 44, no. 10, p. 106401, 2005.

[43] G. Boccignone, A. Chianese, and A. Picariello, “Small target detection using wavelets,” *Pattern Recognition, International Conference on*, vol. 2, p. 1776, 1998.

[44] S. Kim and J. Lee, “Scale invariant small target detection by optimizing signal-to-clutter ratio in heterogeneous background for infrared search and track,” *Pattern Recogn.*, vol. 45, no. 1, pp. 393–406, Jan. 2012.

[45] B. Ristic, S. Arumluampalam, and N. Gordon, *Beyond the Kalman Filter-Particle Filters for Tracking Applications*. Boston, MA: Artech House, 2004.

[46] M. Rutten, N. Gordon, and S. Maskell, “Recursive track-before-detect with target amplitude fluctuations,” *Radar, Sonar and Navigation, IEE Proceedings -*, vol. 152, no. 5, pp. 345 – 352, Oct. 2005.

[47] M. Rutten, B. Ristic, and N. Gordon, “A comparison of particle filters for recursive track-before-detect,” in *Information Fusion, 2005 8th International Conference on*, vol. 1, Jul. 2005, p. 7 pp.

[48] M. Morelande and B. Ristic, “Signal-to-noise ratio threshold effect in track before detect,” *Radar, Sonar Navigation, IET*, vol. 3, no. 6, pp. 601 –608, Dec. 2009.

[49] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking,” *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, 2001.

[50] V. Samson, F. Champagnat, and J.-F. Giovannelli, “Point target detection and subpixel position estimation in optical imagery,” *Applied Optics*, vol. 43, no. 2, pp. 257–263, Jan. 2004.

[51] S. Cheung and C. Kamath, “Robust techniques for background subtraction in urban traffic video,” in *Video Communications and Vision Processing*. San Jose, Ca: SPIE Electronic Imaging, Jan. 2004.

[52] T. Bouwmans, “Subspace learning for background modeling - a survey,” in *Recent Patents on Computer Science*, vol. 2, no. 3, Nov. 2009, pp. 223–234.

[53] B. V. T. Bouwmans, F. El Baf, “Background modeling using mixture of Gaussians for foreground detection - a survey,” in *Recent Patents on Computer Science*, vol. 1, no. 3, Nov. 2008, pp. 219–237.

[54] ——, “Statistical background modeling for foreground detection: A survey,” in *Handbook of Pattern Recognition and Computer Vision*, vol. 4, no. 2. World Scientific Publishing, Jan. 2010, pp. 181–199.

[55] T. D. C. Wren, A. Azarbayejani and A. Pentland, “Pfinder: real-time tracking of the human body,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, 1997, pp. 780–785.

[56] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, “Detecting objects, shadows and ghosts in video streams by exploiting color and motion information,” in *Image Analysis and Processing, 2001. Proceedings. 11th International Conference on*, Sep. 2001, pp. 360 –365.

[57] C. S. N. McFarlane, “Segmentation and tracking of piglets in images,” in *British Machine Vision and Applications*, 1995, pp. 187–193.

[58] C. Stauffer and W. Grimson, “Adaptive background mixture models for real-time tracking,” *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 2, p. 2246, 1999.

[59] P. W. Power and J. A. Schoonees, “Understanding background mixture models for foreground segmentation,” in *Proc. Image and Vision Computing New Zealand*, Auckland, Nov. 2002, pp. 267–271.

[60] H. Kim, R. Sakamoto, I. Kitahara, T. Toriyama, and K. Kogure, “Robust silhouette extraction technique using background subtraction,” in *10th Meeting on Image Recognition and Understanding*, Hiroshima, Japan, Jul. 2007.

[61] Z. Zivkovic, “Improved adaptive Gaussian mixture model for background subtraction,” in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2, Aug. 2004, pp. 28 – 31 Vol.2.

[62] J. Cheng, J. Yang, Y. Zhou, and Y. Cui, “Flexible background mixture models for foreground segmentation,” *Image Vision Comput.*, vol. 24, no. 5, pp. 473–482, May 2006.

[63] A. Shimada, D. Arita, and R.-i. Taniguchi, “Dynamic control of adaptive mixture-of-Gaussians background model,” in *Proceedings of the IEEE International Conference on Video and Signal Based Surveillance*, ser. AVSS ’06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 5–.

[64] R. Tan, H. Huo, J. Qian, and T. Fang, “Traffic video segmentation using adaptive-k gaussian mixture model,” in *Proceedings of the 2006 Advances in Machine Vision, Image Processing, and Pattern Analysis international conference on Intelligent Computing in Pattern Analysis/Synthesis*, ser. IWICPAS’06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 125–134.

[65] V. Morellas, I. Pavlidis, and P. Tsiamyrtzis, “DETER: detection of events for threat evaluation and recognition,” *Mach. Vision Appl.*, vol. 15, no. 1, pp. 29–45, Oct. 2003.

[66] D. Lee, “Effective Gaussian mixture learning for video background subtraction,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 5, pp. 827 –832, May 2005.

[67] Y. Zhang, Z. Liang, Z. Hou, H. Wang, and M. Tan, “An adaptive mixture Gaussian background model with online background reconstruction and adjustable foreground mergence time for motion segmentation,” in *Industrial Technology, 2005. ICIT 2005. IEEE International Conference on*, Dec. 2005, pp. 23 – 27.

[68] M. Amintoosi, F. Farbiz, M. Fathy, M. Analoui, and N. Mozayani, “QR decomposition-based algorithm for background subtraction,” in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 1, Apr. 2007, pp. 1093–1096.

[69] A. Lepisk, “The use of optic flow within background subtraction,” Ph.D. dissertation, Royal Institute of Technology, Nada, Sweden, Jan. 2005.

[70] H. Wang and D. Suter, “A re-evaluation of mixture-of-Gaussian background modeling,” 2005.

[71] J. Lindström, F. Lindgren, K. Åström, J. Holst, and U. Holst, “Background and foreground modeling using an online EM algorithm,” in *IEEE International Workshop on Visual Surveillance*, 2006.

[72] A. Elgammal, D. Harwood, and L. S. Davis, “Non-parametric model for background subtraction,” in *IEEE 7th International Conference on Computer Vision*, Kerkyra, Greece, Sep. 1999.

[73] ——, “Non-parametric model for background subtraction,” in *6th European Conference on Computer Vision*, Dublin, Ireland, Jul. 2000.

[74] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis, “Background and foreground modeling using nonparametric kernel density estimation for visual surveillance,” *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1151 – 1163, Jul. 2002.

[75] A. Bessell, B. Ristic, A. Farina, X. Wang, and M. Arulampalam, “Error performance bounds for tracking a manoeuvring target,” in *Information Fusion, 2003. Proceedings of the Sixth International Conference of*, vol. 2, 2003, pp. 903 – 910.

[76] Z. Kalal, K. Mikolajczyk, and J. Matas, “Forward-backward error: Automatic detection of tracking failures.” in *ICPR’10*, 2010, pp. 2756–2759.

[77] H. Wu and A. C. Sankaranarayanan, “In situ evaluation of tracking algorithms using time reversed chain,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[78] F. Bashir and F. Porikli, “Performance evaluation of object detection and tracking systems,” in *Performance Evaluation of Tracking and Surviellance*, 2006.

[79] C. DeCusatis, J. Enoch, V. Lakshminarayana, G. Li, C. MacDonald, V. Mahajan, and E. V. Stryland, *Handbook of Optics*, 3rd ed., M. Bass, Ed. McGraw Hill, 2010, vol. 4.

[80] C. Shannon, “Communication in the presence of noise,” *Proceedings of the IRE*, vol. 37, no. 1, pp. 10 – 21, Jan. 1949.

[81] R. Olsen, *Remote Sensing from Air and Space*. Monterey, CA: The International Society for Optical Engineering, Nov. 2007.

[82] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*. Torquay, UK: Wiley Publishing, 2009.

[83] P. F. Felzenszwalb and D. P. Huttenlocher, “Distance transforms of sampled functions,” Cornell Computing and Information Science, Tech. Rep., 2004.

[84] S. Brutzer, B. Höferlin, and G. Heidemann, “Evaluation of background subtraction techniques for video surveillance,” in *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011, pp. 1937–1944.

[85] R. Miezianko, “IEEE OTCBVS WS series bench,” Terravic Research Infrared Database, Tech. Rep.

[86] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” in *Sixth Symposium on Operating System Design and Implementation*, Dec. 2004.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Dr. Steve Finney
   National Geospatial Intelligence Agency